SYSTEMS ENGINEERING
Research Center

Prototype of a Graphical CONOPS (Concept of Operations) Development Environment for Agile Systems Engineering

Final Technical Report SERC-2013-TR-030-2

August 30, 2013

Principal Investigator: Dr. Robert Cloutier, Stevens Institute of Technology

Team Members:

Dr. Drew Hamilton, Senior Researcher - Auburn University

Dr. Teresa Zigh, Senior Researcher - Stevens Institute of Technology

Dr. Peter Korfiatis, Research Assistant - Stevens Institute of Technology

Behnam Esfahbod, Research Assistant - Stevens Institute of Technology

Peizhu Zhang, Research Assistant - Stevens Institute of Technology

Patrick Pape, Research Assistant - Auburn University

Jim O’Brian - Auburn University

Sarah Weeks - Auburn University

Contract Number: H98230-08-D-0171 TTO 0025, RT0030a

Report No. 2013-TR-030-2
August 30, 2013
UNCLASSIFIED

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE **30 AUG 2013**	2. REPORT TYPE **Final**	3. DATES COVERED
4. TITLE AND SUBTITLE **Prototype of a Graphical CONOPS (Concept of Operations) Development Environment for Agile Systems Engineering**		5a. CONTRACT NUMBER **H98230-08-D-0171**
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) **Cloutier /Dr. Robert**		5d. PROJECT NUMBER **RT 30a**
		5e. TASK NUMBER **TTO 0025**
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Stevens Institute of Technology Auburn University**		8. PERFORMING ORGANIZATION REPORT NUMBER **SERC-2013-TR-030-2**
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) **DASD (SE)**		10. SPONSOR/MONITOR’S ACRONYM(S)
		11. SPONSOR/MONITOR’S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release, distribution unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
The goal of the research performed under this task was to continue the investigation of graphical 3D gaming environments in the construction of a shared mental model during concept development. A result of the research is an artifact that is a "proof of concept" prototype, the Integrated Concept Engineering Framework (ICEF). The ICEF is intended to provide a 3D virtual guide through the development of a CONOPS, and also to integrate various tools and applications currently in common usage across industry. This integration is a widely-sought capability, one which will enable current CONOPS developers and users the flexibility to import and export analysis parameters and results to and from various familiar and well-used tools. Legacy systems are a fact of life in operational concerns; this prototype is intended to demonstrate interconnectivity on a limited scale between specific simulation and mathematical modeling software packages, via main operational environment that was built using a game development engine. This final report summarizes the work performed toward improving the process of generating a concept of operations through the use of 3D graphical tools. Section-2 of this report contains reports that have been documented from previous phases of this research project that started in July,2009.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**UU**	**140**	

ABSTRACT

This research is a continuation of research initiated in July 2009. From the first research effort, three other efforts emerged. A number of reports have been issued to document the findings of each phase of this research. Those documents are listed in Section 2 of this report. The technical reports can be found on the Systems Engineering Research Center (SERC) website (http://sercuarc.org/).

The goal of the research performed under this task was to continue the investigation of graphical 3D gaming environments in the construction of a shared mental model during concept development. A result of the research is an artifact that is a "proof of concept" prototype, the Integrated Concept Engineering Framework (ICEF). The ICEF is intended to provide a 3D virtual guide through the development of a CONOPS, and also to integrate various tools and applications currently in common usage across industry. This integration is a widely-sought capability, one which will enable current CONOPS developers and users the flexibility to import and export analysis parameters and results to and from various familiar and well-used tools. Legacy systems are a fact of life in operational concerns; this prototype is intended to demonstrate interconnectivity on a limited scale between specific simulation and mathematical modeling software packages, via a main operational environment. This environment was built using a game development engine.

This task was always envisioned as part of a larger CONOPS research agenda. As research progressed, the potential synergies from merging RT 31 with RT 23, and then combining development architecture and strategies with RT 30, became apparent. Some already-developed external interfaces were seen as adjuncts to the activities performed in RT 30 and these interfaces would certainly be useful in the future. By far, the greater synergies in the development effort were in architecture and operational issues – such issues are transparent to the user but vital to successful delivery. Further exploration led to an integrated data-set and application.

This final report contains significant sections from the previous reports, and is intended to summarize the culmination of work performed toward improving the process of generating a concept of operations through the use of 3D graphical tools.

This page intentionally left blank

TABLE OF CONTENTS

FIGURES AND TABLES

1.0 SUMMARY

The Department of Defense (DoD) is vigorously pursuing greater efficiency and productivity in defense spending so that it can continue to provide the armed forces with superior capabilities in an environment of flat defense budgets. Toward that end, the Office of the Secretary of Defense (OSD) has issued new acquisition guidance that places increased emphasis on system engineering early in the lifecycle to balance operational performance with affordability and has established the System Engineering Research Center (SERC) to create the tools and processes needed to execute this guidance. As one of its research areas, the SERC has put forth the notion of a concept engineering system for agile CONOPS Development.

Technical Reports SERC-2009-TR-003 and SERC 2010-TR-007 provided a compelling vision, a feasibility assessment, and an initial process definition for Graphical CONOPS development environment for agile systems engineering. Technical Report SERC-2011-TR-030 detailed the successful integration of several analysis software packages, resulting in an initial prototype which demonstrated a cohesive and easy to use collaborative concept engineering system applicable within the DoD acquisition domain.

This research extends the proof of concept prototype originally dubbed the “CONOPS Engineering System”. This prototype provides a 3D virtual guide intended to assist one assigned to CONOPS development, through the setup of a combat scenario and the use of the Integrated CONOPS Environment Framework (ICEF). Where previous research tasks had investigated data modeling tools and the seamless transfer and manipulation of data from one application to another using Excel, @Risk, and MATLAB, the thrust of this research would be to setup and run an intelligence gathering scenario.

Finally, using the developed prototype, data was collected through surveys, observational evaluations and computer generated records and analyzed using Bayesian hypothesis testing. The data collected showed strong evidence to support the validity of the research hypotheses that such a graphical approach would improve the process of CONOPS development over the more traditional, text based approaches.

2.0 PUBLICATIONS RESULTING FROM THIS RESEARCH

This research has produced a considerable number of technical reports and research papers (both conference papers, and peer reviewed journal papers).

2.1 JOURNAL ARTICLES PRODUCED FROM THIS RESEARCH

Mostashari, A., McComb, S. A., Kennedy, D. M., Cloutier, R., & Korfiatis, P. (2012). Developing a stakeholder-assisted agile CONOPS development process. *Systems Engineering, 15*(1), 1-13. doi: 10.1002/sys.20190

[This article was the second most downloaded Systems Engineering article in 2012, with 1185 downloads]

2.2 CONFERENCE PAPERS PRODUCED FROM THIS RESEARCH

Korfiatis, P. and R. Cloutier. (2013). Using 3D Gaming Technologies to Improve the Concept of Operations (CONOPS) Process. 2013 NDIA Ground Vehicle Systems Engineering and Technology Symposium, Modeling & Simulation, Testing and Validation (MSTV) Mini-Symposium. Troy, MI, August 21-22, 2013

Korfiatis, P., Cloutier, R., Zigh, T. (2012). Graphical CONOPS Development to Enhance Model Based Systems Engineering, *Third International Engineering Systems Symposium, CESUN 2012*, Delft University of Technology, The Netherlands, 18-20 June 2012

2.3 TECHNICAL REPORTS PRODUCED FROM THIS RESEARCH

Cloutier, R. (PI), McComb, S., Deshmukh, A., Zigh, T., Korfiatis, P., Esfahbod, B., Zhang, P., & Hall, K. (2012). *Prototype of a Graphical CONOPS (Concept of Operations) Development Environment for Agile Systems Engineering, Final Technical Report (SERC-2011-TR-030)*

Robert Cloutier (PI), Teresa Zigh, Peter Korfiatis, Behnam Esfahbod, Peizhu Zhang, and John Santanello. (2012). *Graphical CONOPS prototype to demonstrate emerging methods, processes, and tools at ARDEC, Final Technical Report (SERC-2011-TR-031)*

Robert Cloutier (PI), Peter Korfiatis, Kyle Thompson-Bass. (2012). *Communications Effects Server (CES) Model for Systems Engineering Research, Final Technical Report(SERC-2011-TR-023)*

Cloutier, R. (PI), Mostashari, A., McComb, S., Deshmukh, A., Kennedy, D., Korfiatis, P. and Anne Carrigy. (2010). *Investigation of a Graphical CONOPS Development Environment for Agile Systems Engineering, Final Technical Report (SERC-2010-TR-007)*

Robert Cloutier, Robert, Ali Mostashari, Sara McComb, Abhijit Deshmukh, Jon Wade, Deanna Kennedy, and Peter Korfiatis. (2009). *Investigation of a Graphical CONOPS (Concept of Operations) Development for Agile Systems Engineering, Technical Report (SERC-2009-TR-003)*

3.0 INTRODUCTION

It is believed that 3D gaming technologies available today can be used to provide a useful “front end” to the concept engineering process. Selection of the correct game development platform was critical to the implementation.

3.1 USE OF GAMING TECHNOLOGY AS CONDUIT FOR INTEROPERABILITY COMMUNICATIONS

In early 2011, under RT 03, gaming technology was investigated as the core backbone link between all CONOPS-specific functionality – including scenario-building, simulation using various third-party vendor packages, and generating SysML/XML output from vendor offerings already in use by soldiers in the field. To determine which platform to select, a broad range of available gaming environments were examined:

Table 1 Game Development Engines

Torque 2D	Unreal DK	Vicious
Torque 3D	ID Tech (Doom 3)	Open Simulator
Quest 3D	Cry Engineer	C4
Unity	MS-XNA	Gamebryo
Unity Pro	Adobe Flash	Dark Basic
Unreal Engine	Source	Open Simulator

The survey provided a qualitative evaluation of each platform on a number of criteria within several overall categories, as shown below:

<u>Features/Capabilities</u>
- Multiplayer
- 3D/2D representations
- Specific comparative strengths and limitations
- Development languages and physics engines supported

<u>Deployment</u>
- Client-Server capability
- Web, PC, Mac supportable
- Minimum CPU and RAM required
- Video card
- Minimum bandwidth

<u>Compatibility with Open Source</u>
- Source code
- Open source components

- Open interfaces

<u>Cost:</u>
- per seat
- to deploy
- license specifications

The evaluation of the software packages/environments along these dimensions are shown in Figure 1.

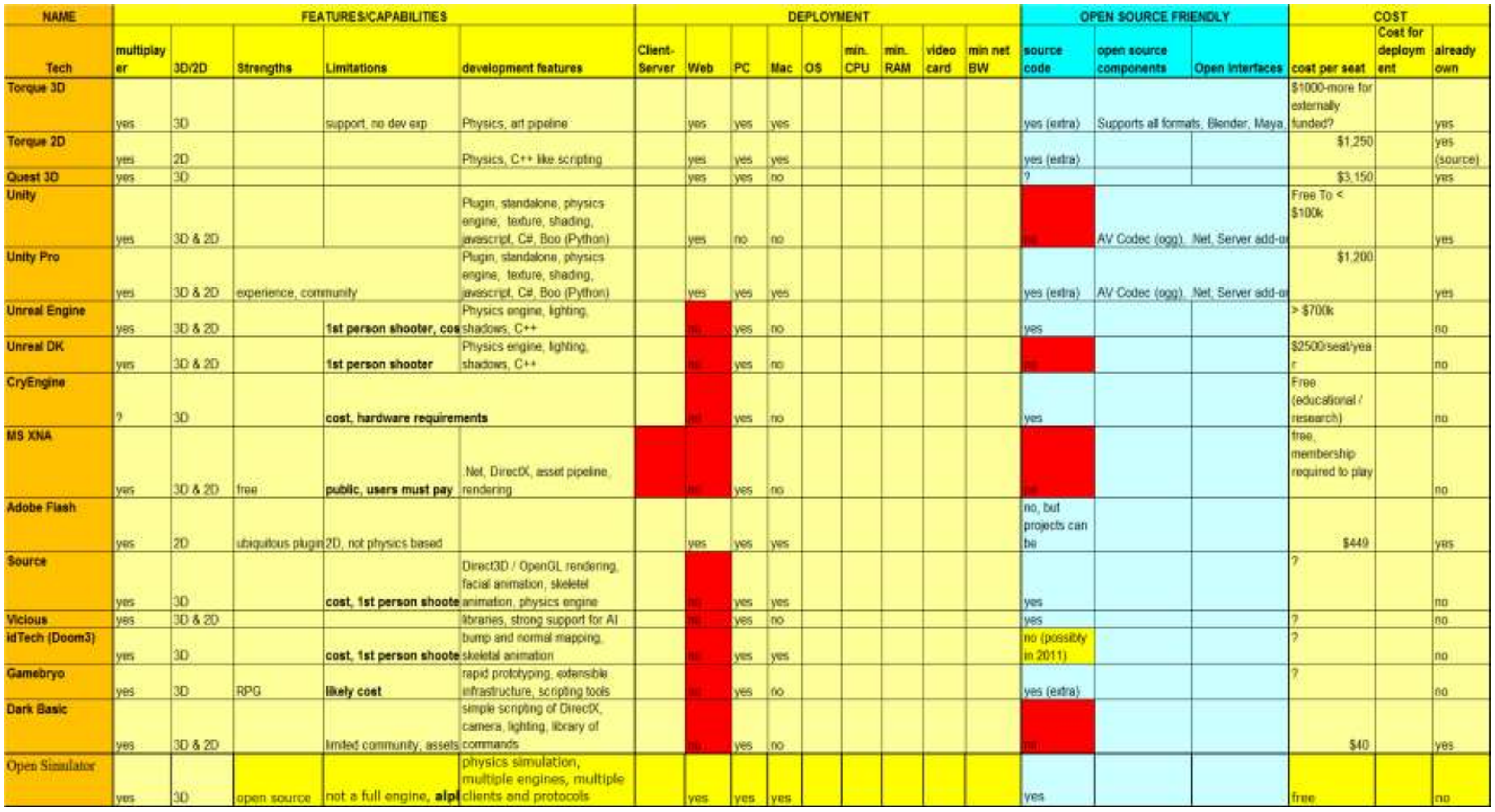
Of all the criteria evaluated, several dominated the decision-making process; most of these concerned development and deployment. These included (in no particular order):

- an active and responsive user community,
- the ability to port to different platforms easily,
- the ability to easily support multiple developers,
- providing code control (though this is not a production environment),
- supporting a diversity of programming languages transparently, and
- the ability to either have or incorporate open source components.

In today's environment of flat defense budgets, cost is also a factor, although site-wide and server licenses may help mitigate concerns that per seat licenses may incur.

Although not stated as one of the "critical" components of the decision-making process, the availability of scalable 3D models was also crucial. The applications will be operating in (and as) a visually-based immersive environment; having the models and simulation as realistic as possible will help increase the probability of acceptance and usage by the eventual field users. 3D models can also have a considerable cost factor. For the initial RT-30 task, the group utilized 3D models that were found at no cost. For RT-31a, several models needed to be purchased, to represent actual soldiers carrying relatively realistic-looking weaponry. Although the selected platform does have extensive libraries of 3D models, most are available at a nominal fee. Those models requiring animation almost always cost more money.

Most of the platforms also had other limitations, another factor when selecting the platform – cost and point-of-view being two major considerations.



NAME	FEATURES/CAPABILITIES					DEPLOYMENT									OPEN SOURCE FRIENDLY			COST		
Tech	multiplayer	3D/2D	Strengths	Limitations	development features	Client-Server	Web	PC	Mac	OS	min. CPU	min. RAM	video card	min net BW	source code	open source components	Open Interfaces	cost per seat	Cost for deployment	already own
Torque 3D	yes	3D		support, no dev exp	Physics, art pipeline		yes	yes	yes						yes (extra)	Supports all formats, Blender, Maya,		$1000-more for externally funded?		yes
Torque 2D	yes	2D			Physics, C++ like scripting		yes	yes	yes						yes (extra)			$1,250		yes (source)
Quest 3D	yes	3D					yes	yes	no						?			$3,150		yes
Unity	yes	3D & 2D			Plugin, standalone, physics engine, texture, shading, javascript, C#, Boo (Python)		yes	no	no						no	AV Codec (ogg), .Net, Server add-o		Free To < $100k		yes
Unity Pro	yes	3D & 2D	experience, community		Plugin, standalone, physics engine, texture, shading, javascript, C#, Boo (Python)		yes	yes	yes						yes (extra)	AV Codec (ogg), .Net, Server add-o		$1,200		yes
Unreal Engine	yes	3D & 2D		**1st person shooter, cos**	Physics engine, lighting, shadows, C++		no	yes	no						yes			> $700k		no
Unreal DK	yes	3D & 2D		**1st person shooter**	Physics engine, lighting, shadows, C++		no	yes	no						no			$2500/seat/year		no
CryEngine	?	3D		**cost, hardware requirements**			no	yes	no						yes			Free (educational / research)		no
MS XNA	yes	3D & 2D	free	**public, users must pay**	.Net, DirectX, asset pipeline, rendering	🟥	no	yes	no						no			free, membership required to play		no
Adobe Flash	yes	2D	ubiquitous plugin	2D, not physics based			yes	yes	yes						no, but projects can be			$449		yes
Source	yes	3D		**cost, 1st person shoote**	Direct3D / OpenGL rendering, facial animation, skeletel animation, physics engine		no	yes	yes						yes			?		no
Vicious	yes	3D & 2D			libraries, strong support for AI		no	yes	no						yes			?		no
idTech (Doom3)	yes	3D		**cost, 1st person shoote**	bump and normal mapping, skeletal animation		no	yes	yes						no (possibly in 2011)			?		no
Gamebryo	yes	3D	RPG	**likely cost**	rapid prototyping, extensible infrastructure, scripting tools		no	yes	no						yes (extra)			?		no
Dark Basic	yes	3D & 2D		limited community, assets	simple scripting of DirectX, camera, lighting, library of commands		no	yes	no						no			$40		yes
Open Simulator	yes	3D	open source	not a full engine, **alpl**	physics simulation, multiple engines, multiple clients and protocols		yes	yes	yes						yes			free		no

Figure 1 Evaluation of Serious Gaming Technologies

3.2 Final Platform Selection

As can be seen in Figure 1, many of the investigated platforms have major drawbacks (shown in red). Chief among these was their inability to deploy on the Web. A secondary consideration for this phase of the research task is the ability of the tool to interface with open source code and components.

The selected platform was Unity 3D Pro. Being more intuitive, the learning curve for developers was found to be less daunting than that of most of the other platforms, and the facility to develop and deploy components was relatively easily-acquired.

Unity 3D Pro has an Asset Server which acts as a central code storage and a rudimentary code control mechanism. It has a rich library of models, environments, scripts, and other development components available, either free or at a nominal cost.

Unity 3D Pro supports a number of programming languages: C#, Boo (Python), and JavaScript. The Unity physics engine supports movement, collision and gravity for solid objects, and users can modify textures/meshes. This ability will be critical if terrain generation from various USGS databases is to be evaluated.

Unity 3D Pro has a large user community which is extremely responsive to posted questions, and a forum containing posted solutions to many commonly-found problems or desired effects. As this research task was focused mainly on interfaces between 3rd-party software, the research team did not find solutions in user community resources for these tasks, however the resources did help when implementing some of the more complex model representations and movement.

Finally, using Google Trends (http://www.google.com/trends/) , which tracks the trend of search queries, showed a steady upward trend for those searching for information on Unity 3D. The decision to adopt Unity Pro in early 2011 seems to be supported by the continued trending Google data. Figure 2 demonstrates that since early 2011, the Unity Pro game development platform has continued to trend upward (graph collected in late July 2013).

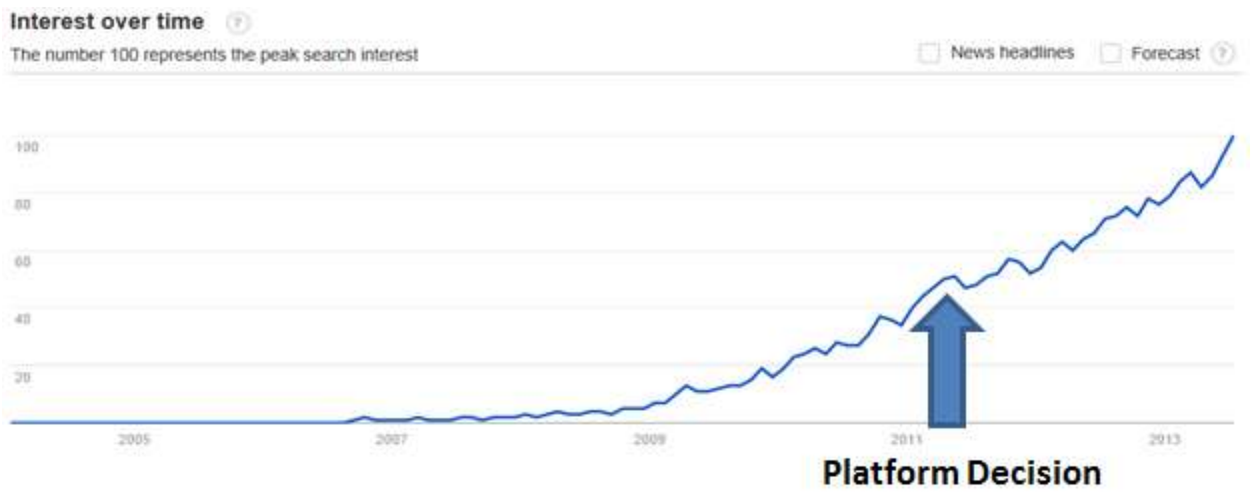


Figure 2 Google Trends for "Unity 3D"

3.3 Database Selection and Other Dependencies

When it came time to select a database to manage multiple domains in a single code set, there were a number of considerations that were necessary to keep in mind. First, Unity is based on Mono Project[1], an open source implementation of the .Net framework. Therefore, it is possible to use common .Net libraries directly in Unity. Mono allows developers to create cross-platform applications easily, using C# and the Common Language Runtime (CLR) that is binary compatible with .Net. Mono runs on Linux, MS-Windows, Mac OS X, BSD, Sun Solaris, Nintendo Wii, Sony PlayStation 3, and Apple iPhone. It will also run on x86, x86-64, IA64, PowerPC, SPARC(32), ARM, Alpha, s390, s390x (32 and 64 bits). The use of Mono and the CLR means that any language that can compile to pure Intermediate Language (IL, sometimes referenced as CIL) should work under Mono. Some Mono-compatible compilers include C#, Python, Java, Scala, Boo, Visual Basic .Net, and Cobra. IL itself is in a binary format and is not readable by humans.

The next dependency was desire to use as much free software as possible. Finding from RT23 showed that one of the major stumbling blocks for use as researchers, and for the Army was the licensing cost of the $50,000 core software on which the Communication Effects Server (QualNet) was built. So, the selection of Apache CouchDB[2], an open source database project developed in the Erlang programming language with a web-based (http) API provided numerous benefits. Apache CouchDB was chosen for several reasons:

1. Erlang is used to build massively scalable real-time systems which require high availability and reliability. Erlang is extensively used in banking, telecom, e-commerce, computer telephone and instant messaging applications.

[1] http://mono-project.com/
[2] http://couchdb.apache.org/

2. Documents are the primary unit of data in Couch DB, and can consist of any number of fields and attachments – this enables the developer to associate 3D models with extensive object information in an encapsulated, efficient fashion.

3. The CouchDB document update model is lockless and optimistic. Authors using client applications save changes to the database - if another client editing the same document at the same time saves their changes first, the author gets an edit conflict error when saving. This gives the author the chance to evaluate the other client's changes, and either accept them or update the document and re-try the update.

4. The database is always in a consistent state; CouchDB will never overwrite committed data or associated structures.

5. CouchDB enables efficient building of views, because the data is stored in semi-structured documents, rather than spread across numerous records and tables. This is of particular interest to this research task.

3.3 Development Process

At the onset of RT 30, the research team was recommending that the Rational Unified Process (RUP) be used as a framework for the ICEF software development effort. Figure 3 shows the basic phases of RUP, and the effort required in each phase by development activities.

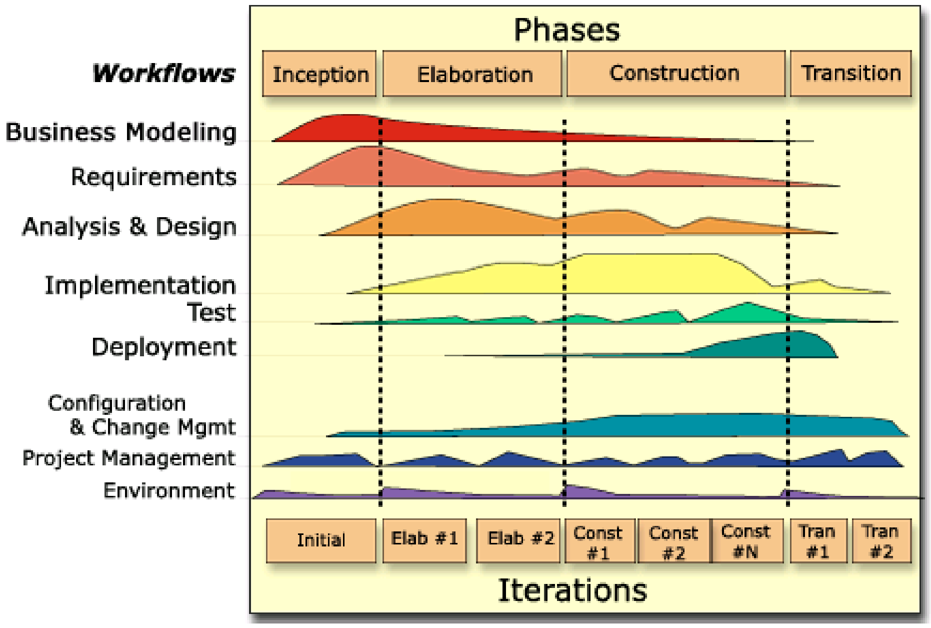


Figure 3 Rational Unified Process for software development

Following RUP on software projects has been shown to lead to a large number of software development successes, however in the case of this project, it proved to be ineffective. The primary reason for its discontinued use was due to the academic environment the research team was working within. As will be detailed below, there were a number of lessons learned when developing a software product using a primarily student-based workforce.

To replace RUP, the research team looked towards a Spiral development type effort. Whereas the initial phases of RT 30 saw software releases every other week which integrated the most stable form of improvements made to ICEF, this phase of the research adopted an agile development approach in which releases were made on a monthly basis, coinciding with the monthly progress report. Each release was made available to the sponsor via the SERC Sakai repository, after testing by technical and non-coding members of the research team.

During the development cycle, new functionality was implemented while the previous release capabilities were tested by both programmers and non-programmers on the research team.

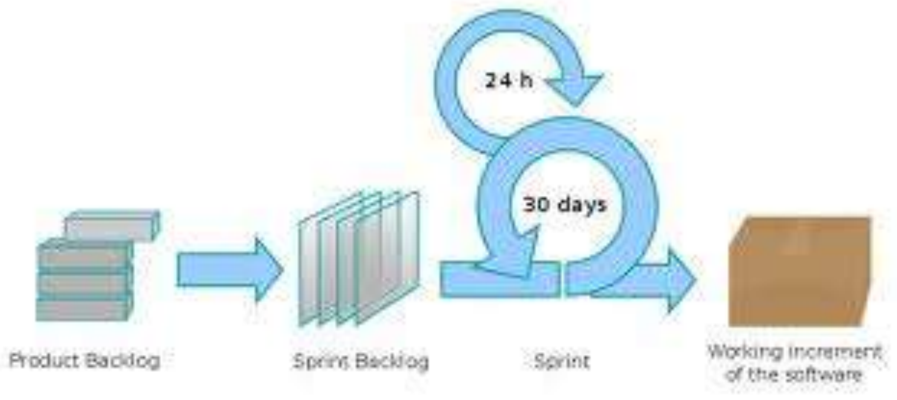
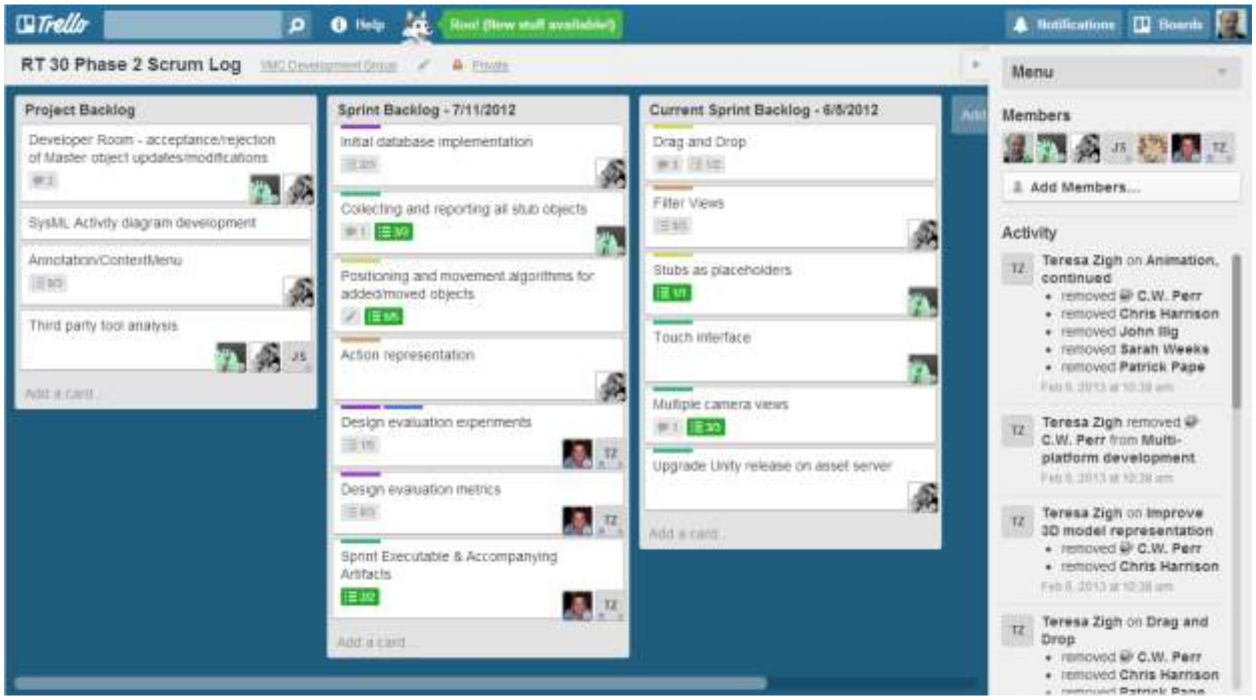


Figure 4 Agile development cycle for monthly releases

3.4 PROJECT COORDINATION

Throughout this work, each phase of research has involved faculty and graduate students across multiple universities. During the early stages of this endeavor, the development process of ICEF and its precursors was managed through weekly meetings. While these meetings were effective, the pace of development often required more frequent interaction between researchers. Distributed development often presents unique challenges for software engineers and the environment encountered in academia can exacerbate these difficulties. Students and faculty often keep irregular hours which can change from week to week. Additionally, the amount and intensity of academic workload can be unevenly spread across a semester, leading to variability in available time to progress software development efforts.

To facilitate better communication, status tracking and visibility surrounding ICEF development, the research team used the online project management tool Trello, developed by Fog Creek Software throughout the later stages of RT30/31.



RT 30 Phase 2 Scrum Log

Project Backlog
- Developer Room - acceptance/rejection of Master object updates/modifications
- SysML Activity diagram development
- Annotation/ContextMenu
- Third party tool analysis

Sprint Backlog - 7/11/2012
- Initial database implementation
- Collecting and reporting all stub objects
- Positioning and movement algorithms for added/moved objects
- Action representation
- Design evaluation experiments
- Design evaluation metrics
- Sprint Executable & Accompanying Artifacts

Current Sprint Backlog - 6/6/2012
- Drag and Drop
- Filter Views
- Stubs as placeholders
- Touch interface
- Multiple camera views
- Upgrade Unity release on asset server

Members

Activity

Teresa Zigh on Animation, continued
- removed C.W. Perr
- removed Chris Harrison
- removed John Ilig
- removed Sarah Weeks
- removed Patrick Pape

Feb 8, 2013 at 10:38 am

Teresa Zigh removed C.W. Perr from Multi-platform development

Feb 8, 2013 at 10:38 am

Teresa Zigh on Improve 3D model representation
- removed C.W. Perr
- removed Chris Harrison

Feb 8, 2013 at 10:38 am

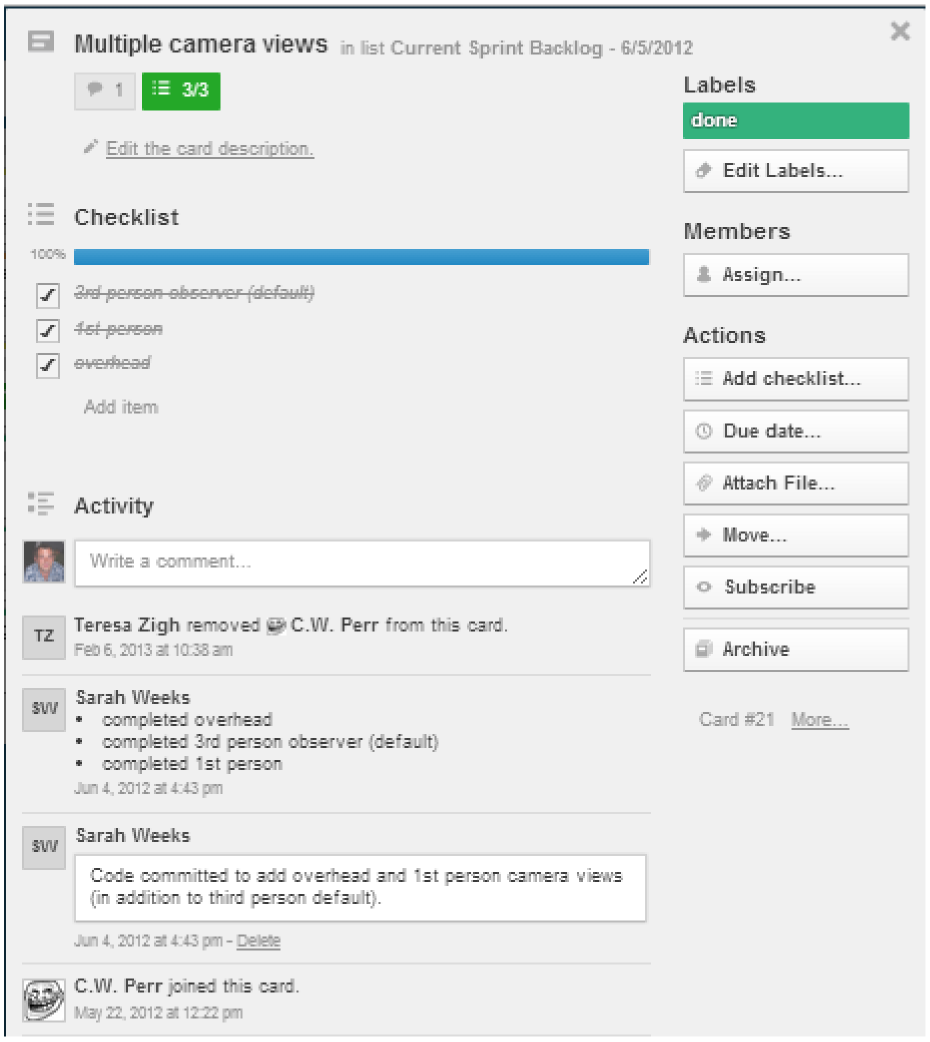
Teresa Zigh on Drag and Drop
- removed C.W. Perr
- removed Chris Harrison
- removed Patrick Pape

Figure 5 Trello Workspace

Trello, as seen in Figure 5, is a cross platform web-based tool inspired by the Kanban paradigm for managing projects. Each project was represented by its on board, with each board containing a number of lists. Each list has cards attached to them, representing major capabilities to be developed for ICEF. Using Trello to implement the agile process shown in Figure 4, each list was designated as a scheduled sprint, and one was labeled Backlog. Figure 5 represents a Trello board with three lists:

- Current Sprint Backlog: A list of capabilities under development during the current monthly sprint.
- Sprint Backlog: A list of capabilities planned for development during the next month's sprint.
- Project Backlog: A complete list of remaining capabilities planned for implementation during the project.

Most of the time, there would be more lists representing 2-4 schedule sprints. Shown in Figure 6, each card represents a specific capability for implementation, and provided the developers and project manager an opportunity to include additional description of tasks, add checklists (denoting lower level capabilities), insert comments, and attach files. Color coding each card allowed the team to communicate development progress at a moment's glance, and labelling of tasks was used to indicate priority among different capabilities.


Multiple camera views in list Current Sprint Backlog - 6/5/2012

Edit the card description.

Checklist

100%

~~3rd person observer (default)~~

~~1st person~~

~~overhead~~

Add item

Activity

Write a comment...

Teresa Zigh removed C.W. Perr from this card.
Feb 6, 2013 at 10:38 am

Sarah Weeks
- completed overhead
- completed 3rd person observer (default)
- completed 1st person

Jun 4, 2012 at 4:43 pm

Sarah Weeks

Code committed to add overhead and 1st person camera views (in addition to third person default).

Jun 4, 2012 at 4:43 pm - Delete

C.W. Perr joined this card.
May 22, 2012 at 12:22 pm

Figure 6 Trello card contents

Nearing the completion of a sprint, the project manager had a clear indication of which capabilities would be fully implemented. Upon completion of a sprint, any capabilities that required further development could be moved to the next sprint's list, and completed capabilities were archived for future use in reporting progress and work breakdown structure to the research sponsor. The Sprint Backlog list was promoted to Current Sprint Backlog, and a new list of capabilities for the next sprint was assembled from the Project Backlog.

While Trello did not completely replace weekly team meetings, it provided a number of benefits throughout the development of ICEF:

- Allowed for more effective weekly meetings, focusing less on status updates and more on high level design, group problem solving and bug identification.
- Provided the project manager with a graphical interface for project schedule planning.
- Provided distributed team members with an easy to use portal for asynchronous reporting of development progress and potential problem areas.
- Presented the project manager with a clear view of work allocation among team members, allowing for workload balancing.
- Promoted communication between team members developing capabilities that would need to be integrated during future sprints.
- Allowed for effortless reporting of monthly progress and project work breakdown structure to research sponsors.
- Provided clear transparency of development progress across the entire team.

In addition to Trello, the team's distributed technology portfolio included Google Forums to facilitate bug reporting. Once a bug was discovered or difficulty was encountered, a team member was able to post to the Forum and have the pertinent details instantly distributed to the entire team. Team members could then email the Forum to provide proposed solutions or advice. This prevented downtime during which development progress would have normally been stalled. It also provided a living record of problems encountered. Finally, Dropbox and the Unity Asset Server were used to distribute the project's software code, most recent builds and supporting documentation across the entire team.

The combination of Trello, Google Forum, Dropbox and the Unity Asset Server provided a highly effective suite of tools to facilitate distributed collaboration across an academic environment. Team members actively participated in asynchronous communication, allowing the team to make efficient use of the synchronous communication opportunities. Feedback among the team regarding these tools and their effect on collaboration was positive, and it is the intention of the principle investigator and project manager to use this collection of tools for future research and development efforts.

4.0 WORK PERFORMED FOR RT30A

This section will focus on the work performed for the RT30a tasking, and then step back to the broader research perspective.

The work performed during this research task includes:

- Continual ICEF architectural development and modeling
- ICEF prototype software development
- Workshop created and executed using sponsor selected personnel to test the ICEF software, validate the CONOPS improvement line of research, gather data for hypothesis testing, and collect user feedback.
- Recommendations for future research and software development efforts for continuation of this research task.

The research team kept sponsors updated to task progress through a number of predetermined meetings, as displayed in Table 2. Additionally, a working meeting was held every week between the members of the research team and a representative from the sponsor to discuss research and development progress, issues and goals.

Table 2: RT30 Sponsor Meeting Schedule

Project Kickoff	5/11/2012
IPR	5/12/2012
Interim Workshop with Sponsor	9/14/2012
Workshop	5/22/2013
Conference for Development Environment	4/16/2013

4.1 ICEF PROTOTYPE

The goal of ICEF is the development of a tool to enable the investigation of RT 030 research questions. The fully-functional prototype of ICEF is meant to act as a proof of concept demonstration of the applicability of leveraging gaming technologies and virtual environments towards solving complex issues faced during concept engineering, and to investigate assertions that the use of graphical CONOPS development tool will result in increased stakeholder participation and improved models.

A high level conceptual architecture for ICEF can be seen in Figure 7.

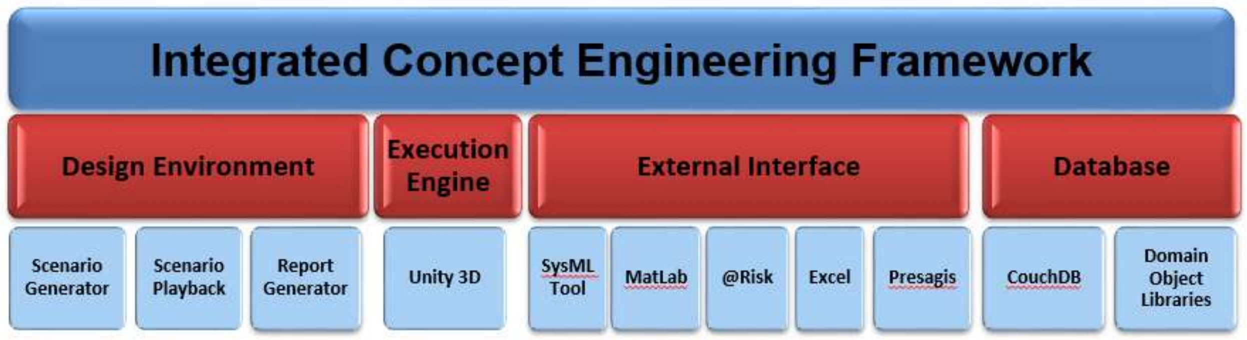


Figure 7 ICEF Conceptual Architecture

The main components of ICEF include:

- Scenario Generator - user interface enabling the creation of the graphical CONOPS
- Databases –for storing primitives; promoting their reuse; storing graphically created scenarios as models
- Report Generator – automatic CONOPS document creation to fulfill current contracting requirements
- Scenario Playback – allowing development and display of animations reflecting the scenario and describing the time components, scenes developed and storyboarded as in movie industry.

4.1.1 Modeling the ICEF Prototype

To guide development of the ICEF Prototype, architecture models were created following a Model-Based Systems Engineering (MBSE) methodology using the System Modeling Language (SysML). Use Case diagrams were developed to describe the nature of interaction between ICEF users and the ICEF system. From here, specific scenarios of possible use of ICEF were explored in Activity diagrams. Based on the Use Case and Activity diagrams, both black box and white box representations of ICEF were developed at a high level. These diagrams are shown below, representing desired ICEF functionality and architecture. As with all software development efforts, ICEF is constantly evolving and updates to the ICEF model were made incrementally. These models are used as design tools for both development and analysis of ICEF programming efforts, allowing the research team to communicate with others and to reason with each other about the architecture of ICEF.

Use Case Diagrams

Figure 8represents the high level interactions between ICEF and the CONOPS author (non-technical user) and primitive developer (advanced user). These interactions allow stakeholders and developers to create a graphical CONOPS model, produce a textual CONOPS artifact and create visualizations based on a CONOPS. Each Use Case is further elaborated in subsequent figures.

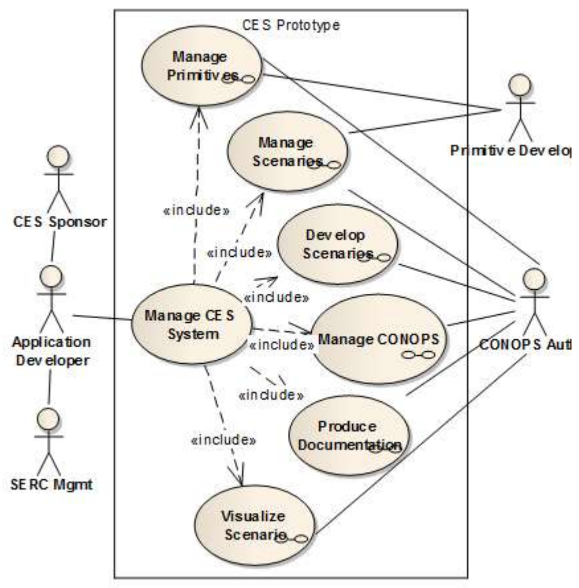


Figure 8 ICEF Prototype Top Level Use Case Diagram

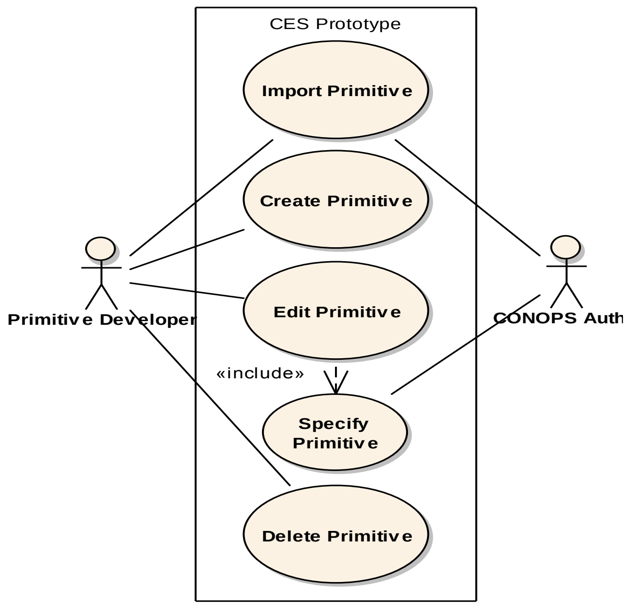


Figure 9 ICEF Prototype Manage Primitives Use Case

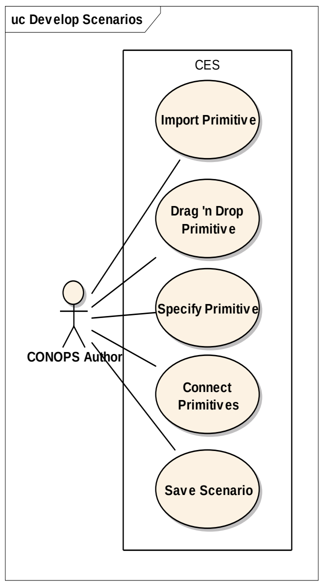


Figure 10 ICEF Prototype Develop Scenarios Use Case

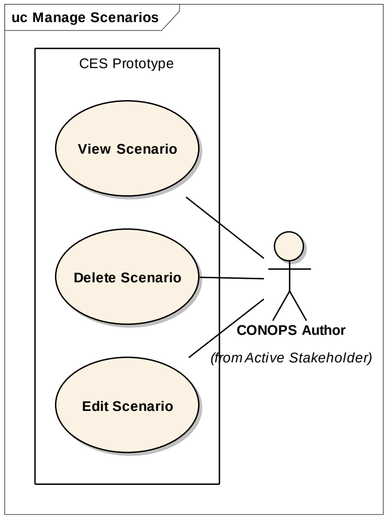


Figure 11 ICEF Prototype Manage Scenarios Use Case

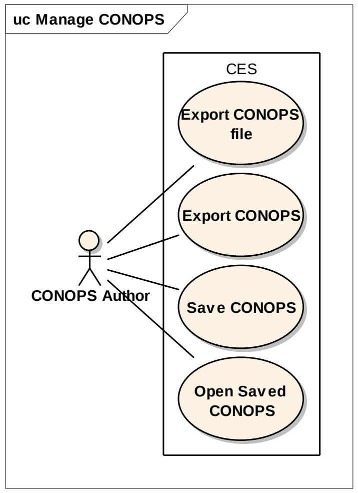


Figure 12 ICEF Prototype Manage CONOPS Use Case

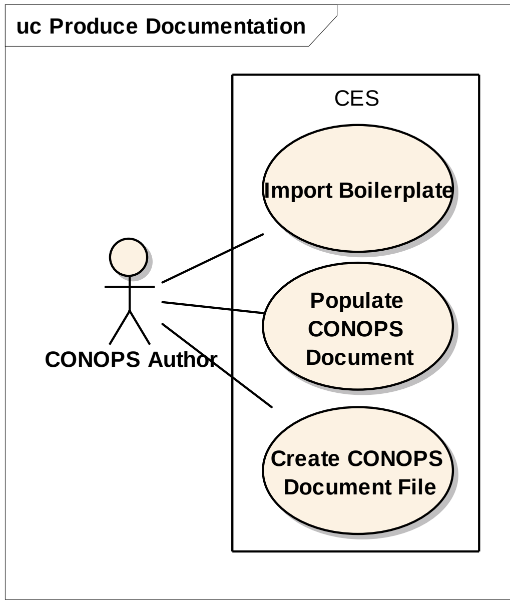


Figure 13 ICEF Prototype Produce Documentation Use Case

In Figure 9, Manage Primitives describes the creation, use and modification of the basic building of a graphical CONOPS. Primitives are defined as the major elements or objects involved in scenarios built within the graphical CONOPS. They can represent people, locations, vehicles, /weapons, computer systems, etc. These primitives reside within a centralized database, allowing primitive developers to create them and make them available for use by CONOPS authors. Each primitive contains attributes that define the properties of the primitive, allowing CONOPS authors to alter primitive characteristics to their needs.

Scenarios, depicted in Figure 10 and Figure 11, involve the linking of individual primitives to tell the story of how a user may interact with a future system. By creating Scenarios, CONOPS authors will be able to describe their needs.

The combination of Scenarios created by the CONOPS author make up the graphical CONOPS model, which can be saved and manipulated in a format that promotes reuse (Figure 12). Since a textual CONOPS is often a requirement for contracting purposes, ICEF will enable CONOPS authors to generate a textual CONOPS from a graphical CONOPS, as depicted in Figure 13.

Activity Diagrams

Activity diagrams give insight into the steps taken by both systems and their users to carry out use cases and provide capabilities to users. Activity diagrams are flow charts, displaying a set of activities carried out (rounded rectangles), by specific actors, systems or subsystems (vertical partitions) leading to the creation of certain artifacts (right angle rectangles). In Figure 14, the Manage Primitives Use Case described above is depicted in terms of the interaction between the ICEF Prototype, the Primitive Developer (advanced user) and an external 3D modeling software tool.

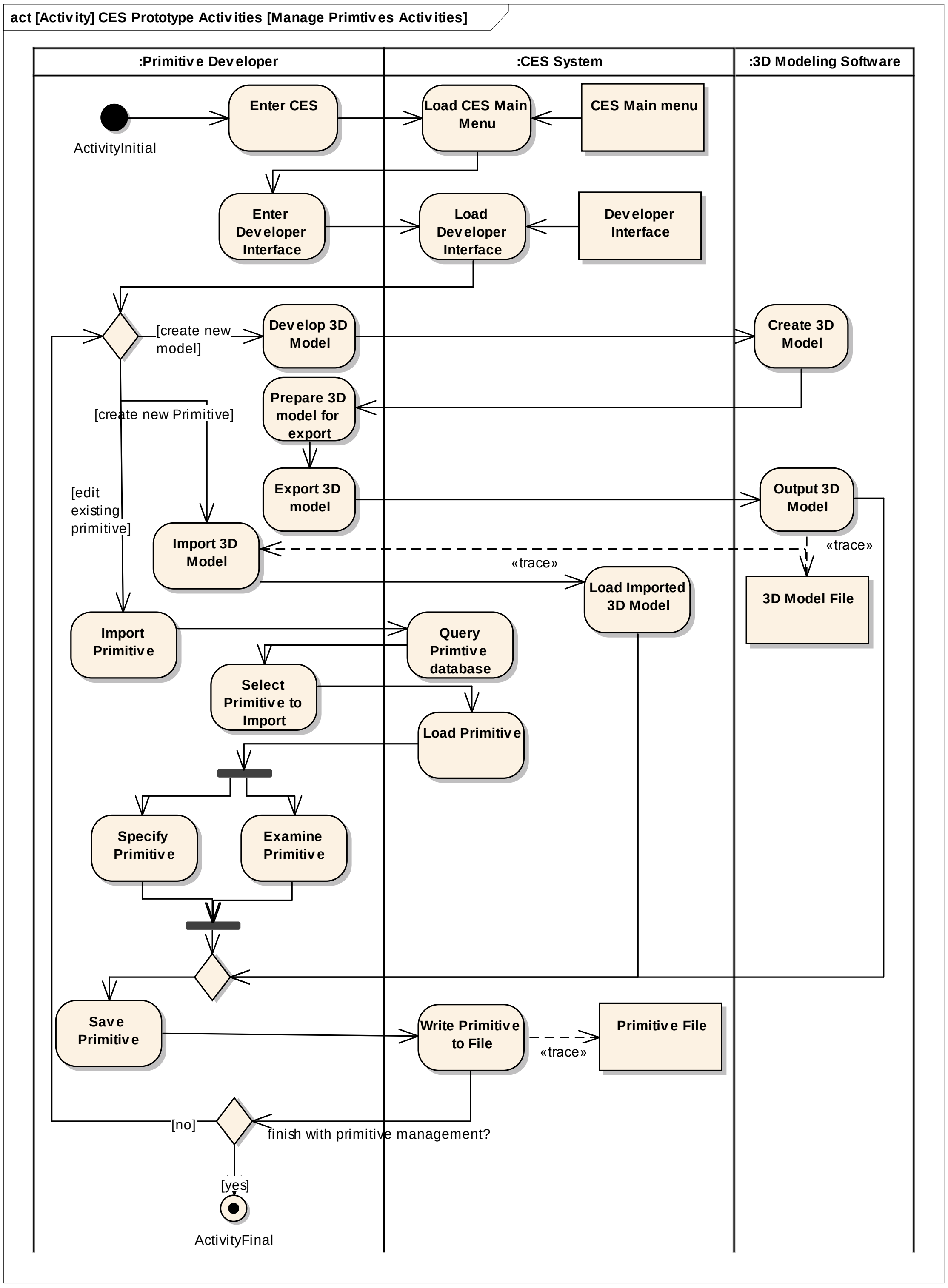


Figure 14 ICEF Prototype Manage Primitives Activities

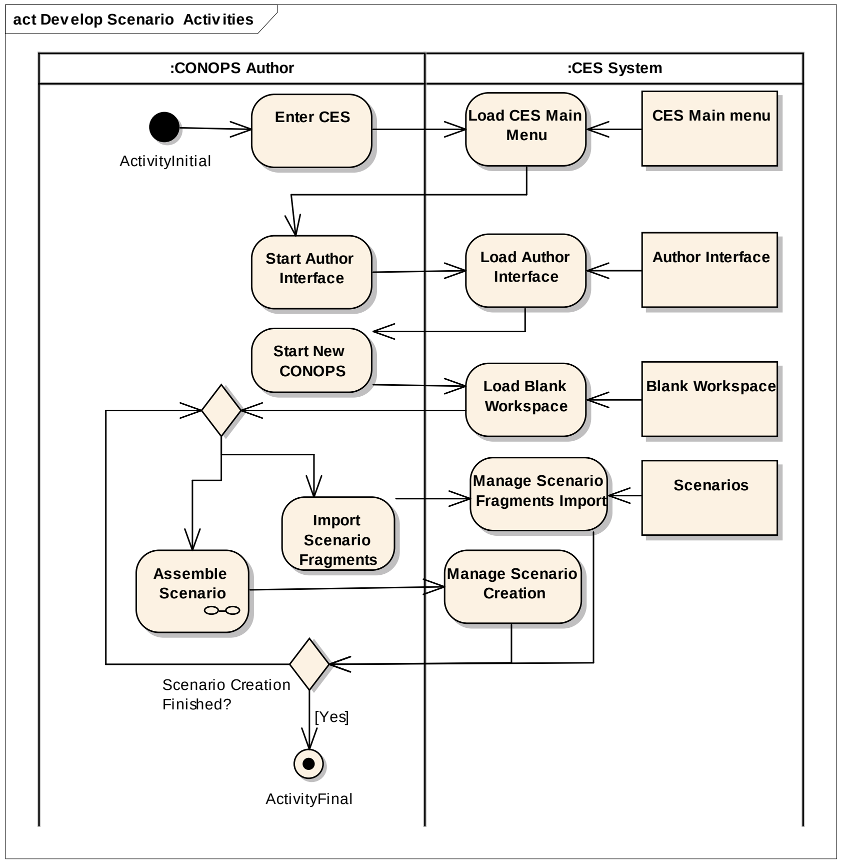


Figure 15 Develop Scenario Activities activity diagram

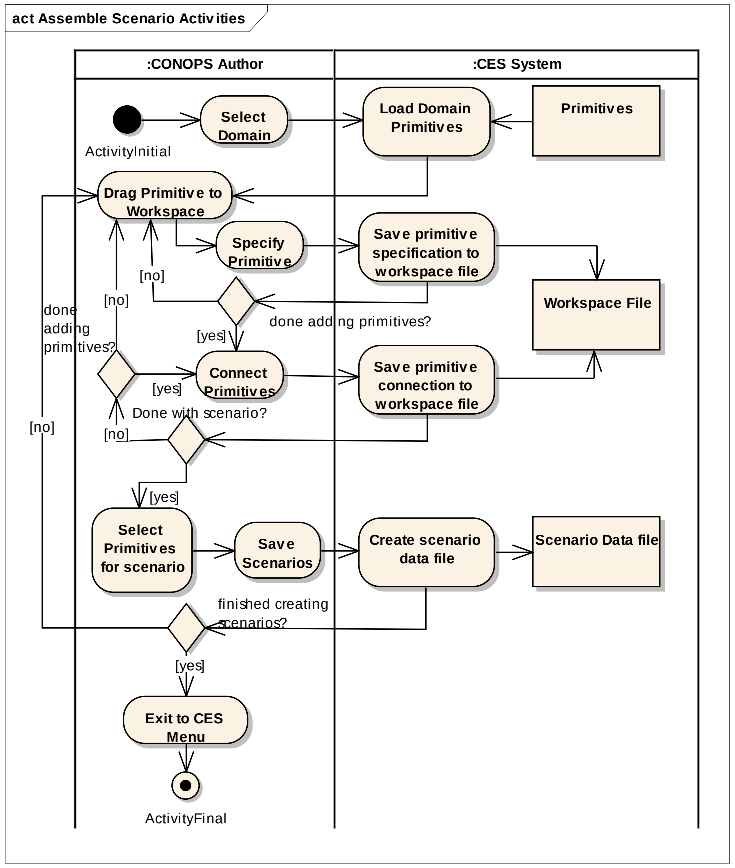


Figure 16 Assemble Scenario Activities

Figure 15 represents typically activities required to carry out the Develop Scenario Use Case displayed in Figure 10. Each activity can be further explored with another Activity diagram, as is the case with Figure 16, which decomposes the Assemble Scenario activity, a task taken on by the CONOPS Author during the development of a scenario.

4.2 Prototype First Release

Screenshots are provided below from the first release of the ICEF prototype. This prototype was provided to sponsor selected evaluators during a workshop, which will be discussed below.

Up entering ICEF, the user is prompted to specify their role. The role the user will assume is based on their purpose in the CONOPS process and their technical expertise. Stakeholders and developers that are creating the CONOPS will enter the Author interface, while those supporting Authors through creation and specification of primitives, scenes and scenarios will enter in the Developer interface.

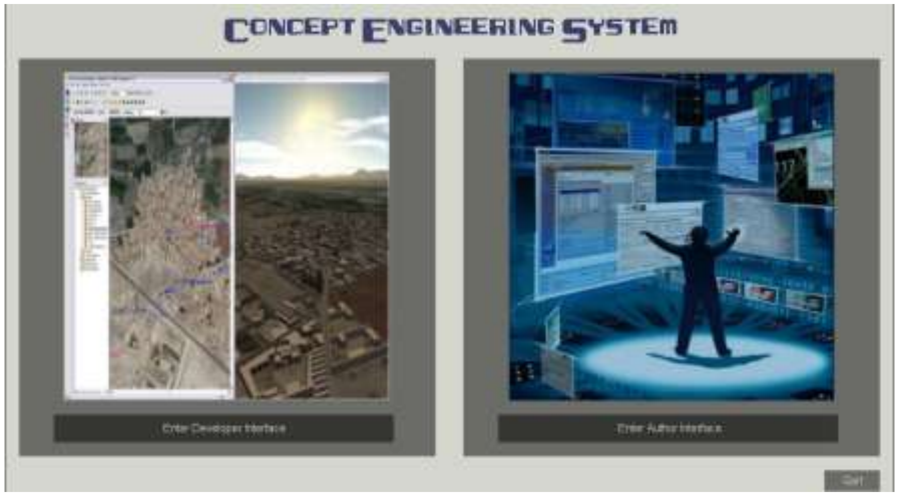
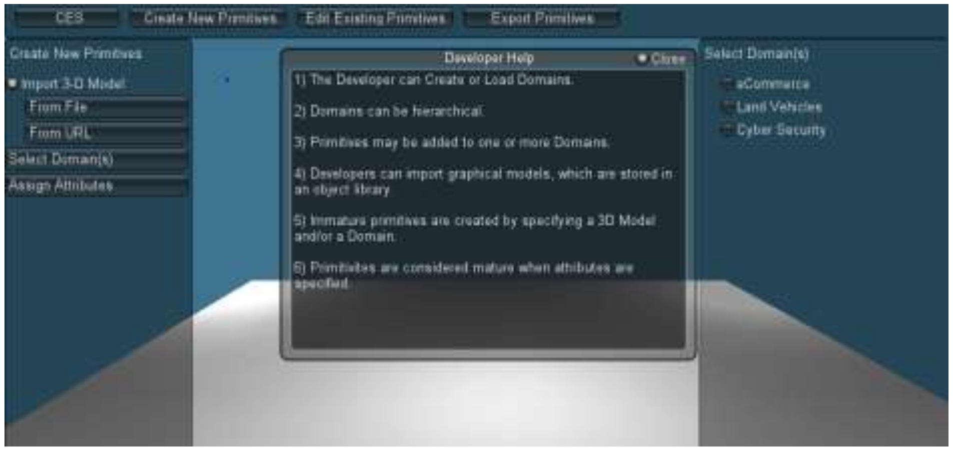


Figure 17 ICEF Prototype Start Screen

The Developer Interface, seen in Figure 18 provides an environment in which advanced users and system developers can create the materials CONOPS authors need to build the graphical CONOPS. This includes the creation or editing of primitives, the assignment of attributes to these primitives and the application of 3D representations to primitives.

When a user first enters the Developer Interface, they are welcomed by a help dialogue, providing them guidance. Whenever the user selects a command in the interface, a new help box is displayed. If the developer is an experienced used of ICEF, the help dialogue boxes can be automatically hidden, and recalled at any time. This feature helps create a work flow for ICEF users, as well as providing an easy to assemble user manual.



1) The Developer can Create or Load Domains.

2) Domains can be hierarchical.

3) Primitives may be added to one or more Domains.

4) Developers can import graphical models, which are stored in an object library.

5) Immature primitives are created by specifying a 3D Model and/or a Domain.

6) Primitiebes are considered mature when attributes are specified.

Figure 18 ICEF Prototype Developer Interface

The Author Interface is displayed in Figure 19, and includes 4 major areas. Dialogue box 1 represents the Author Interface help system, similar to Developer Interface. Area 2 controls the creation and specification of primitives and links, as depicted in Figure 20 and Figure 21. Area 3 acts as the graphical interactive workspace, where primitives can be added, linked, and moved. Finally, along the bottom and right sides of the screen, Area 4 provides controls to move between, create, delete and edit scenes. Each scene has a location, which is shown in the workspace (Area 3), and properties describing the transition between scenes (right hand side bar).

Figure 20 shows the author interface after the user has added four primitives, a man, a woman, a van and a car. From the primitive dialogue box on the upper left side, we can see that adding a primitive to the workspace creates an instantiation of that primitive, inheriting some of the properties of the parent primitive (car has speed profile, acceleration, fuel efficiency as attributes) but allowing the user to specify other properties (specific values for the attributes mentioned above, a specific brand, make, name, etc.).

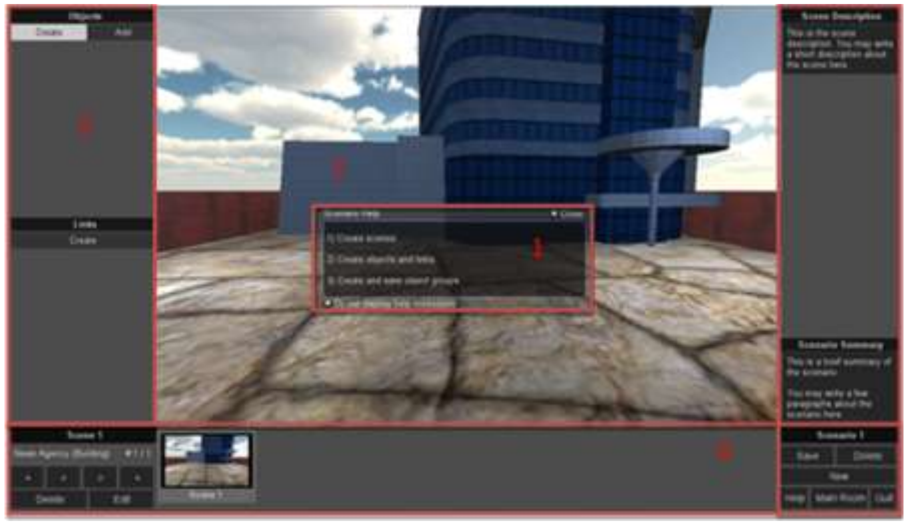


Figure 19 ICEF Prototype Author Interface

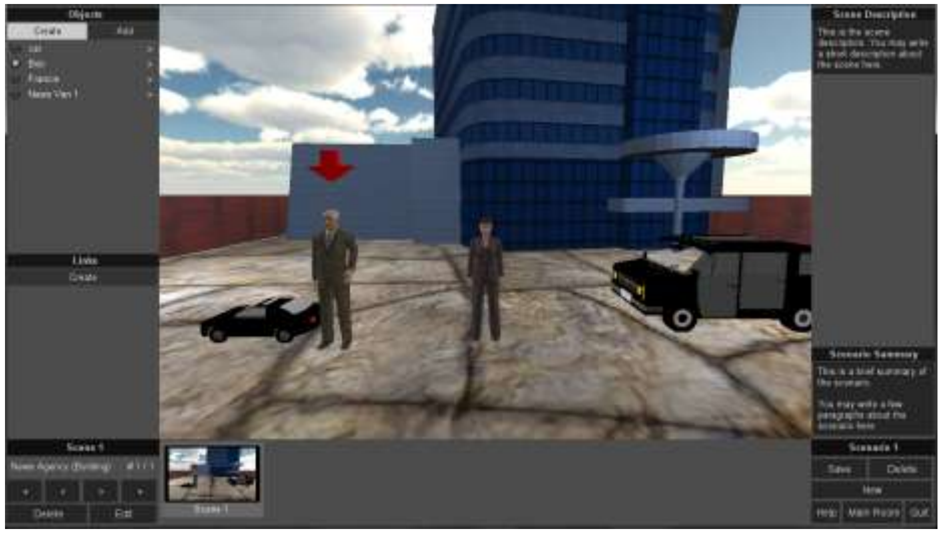


Figure 20 ICEF Prototype: Adding Primitives

Figure 21 displays the Linking functionality of ICEF. Using the Link dialogue box, the user can specify the type of connection that exists between two primitives. In this scene, we can see that the female character Francie is recruiting the male character Bob and Bob is to drive the vehicle. While visible representations are not yet active on in the workspace, future development will allow some visual indicator of linkages. Figure 21 also shows an example pop-up where specific attributes may be entered relating to the link (for example where Bob is driving to, how fast, etc.).

So far, ICEF has assisted the CONOPS author in describing activities that the users and system will carry out in a specific increment of time. A goal of this research is to enable stakeholders to describe their interactions with a system during its operation. Therefore, a temporal component must be present in ICEF to allow representations of operational scenarios over time. Figure 22 shows that the Author Interface organizes scenarios into specific scenes, along the bottom of the screen. Figure 21 was the representation of scene 1; Figure 22 is the representation of scene 5. By allowing authors to create, reorder and move scenes, they are able to better tell the CONOPS story. An advanced goal of ICEF was to allow authors to describe the transitions between scenes in such a way as to allow auto-generation of an animation depicting the possible system/stakeholder interaction scenarios.

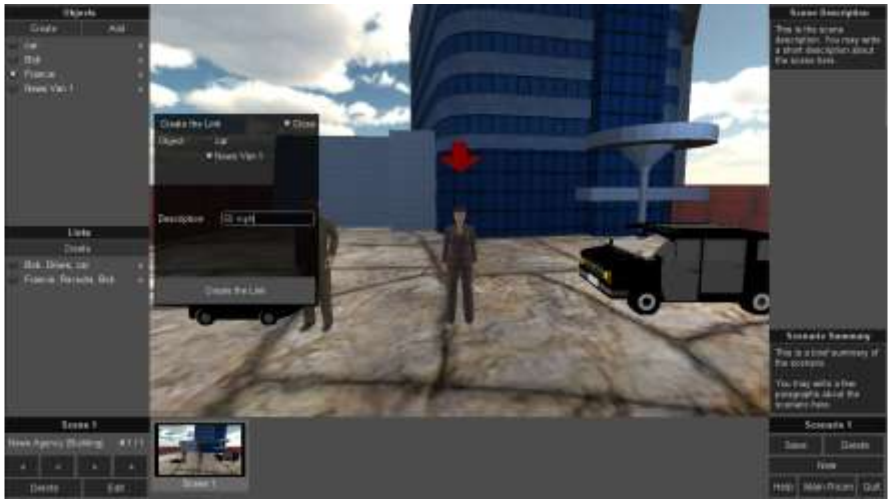


Figure 21 ICEF Prototype: Connecting Primitives

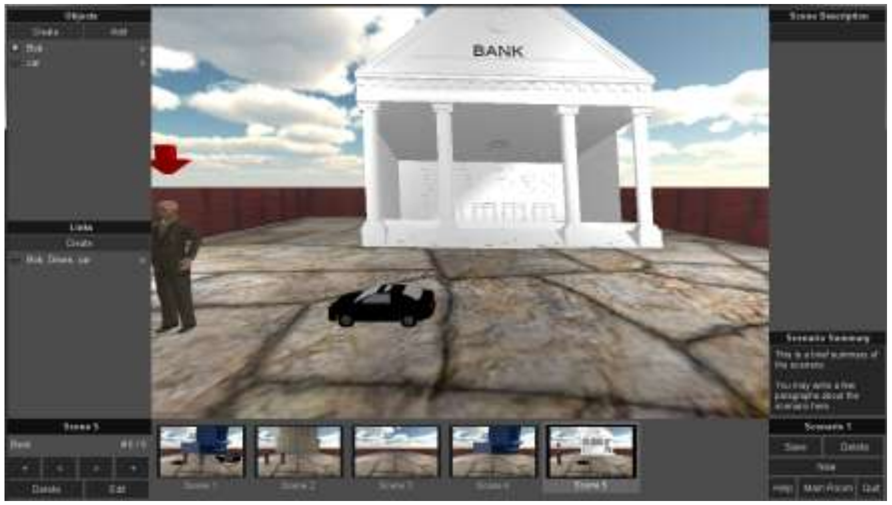


Figure 22 ICEF Prototype: Changing Scenes

4.3 Usability Planning

The research team began to look at what might be involved with a complete usability approach for a virtual environment. That work is detailed in Appendix A of the SERC-2011-TR-030 report. The questionnaires presented may be useful in soliciting information about the usability of the system. Any formal user testing will require approval the university Institutional Review Board (IRB) to ensure fair treatment of the test subjects.

4.4 Extension of Domains/Database Specifics

One goal of this research is the development of a tool that is extendable to multiple domains and situations. With this in mind, the design and implementation of the software relies heavily on an efficient and extensible database representation.

The database is completely separate from the executable developed in the Unity environment (See Figure 23 below). The central database is optional, but would be required for multi-user collaborative authoring. The user's database is required if object and scenario persistence is desired between sessions. A single user need not implement the database if persistence of work performed is not required. The software uses Apache CouchDB management software; which is a document-oriented database package using JSON.

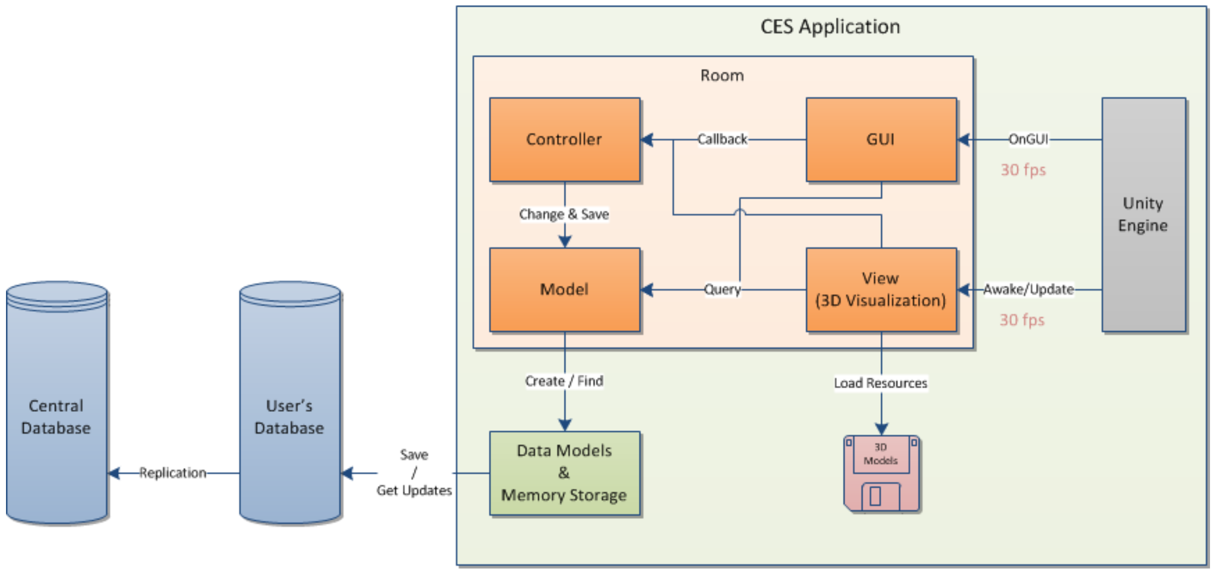


Figure 23 Application Dataflow

The design in this release of ICEF saves each scenario as a separate database entity, thereby disallowing data-sharing between scenarios. This particular limitation was adopted to speed implementation and demonstration of collaboration across users, and is intended to be modified in later versions of ICEF, to allow for sharing of data and objects between scenarios.

The database interface mapping of objects and actions has been developed completely in-house, and is therefore extremely lean and overcomes the limitations of Unity's lack of support for .Net version 3.

Given the persistence of objects, the implementation of a new domain requires only that the 3D named models, the actions, and new locations be added to the database and environment will be immediately usable.

4.4 USE CASE SCENARIO MODELING

The ICEF also provides the ability to model use cases specific to end user's needs. This capability leverages work performed under RT 030 and RT 030a, the ICEF architecture, the modeling of a new domain within the architecture, and the modifications needed to support the new domain.

5.0 ARCHITECTURE OF ICEF SOFTWARE

The ICEF architecture provides flexibility, reusability, and extensibility for this research tool. ICEF subsumes the original CONOPS Navigator capabilities and interfaces, and implicitly generates structured data, which can then be shared and visualized among all stakeholders. Shown below is an overview of the architecture.

The ICEF requires a clear delineation between data and user interaction. This was accomplished by implementing the well know Model-View-Controller pattern, shown in Figure 24.

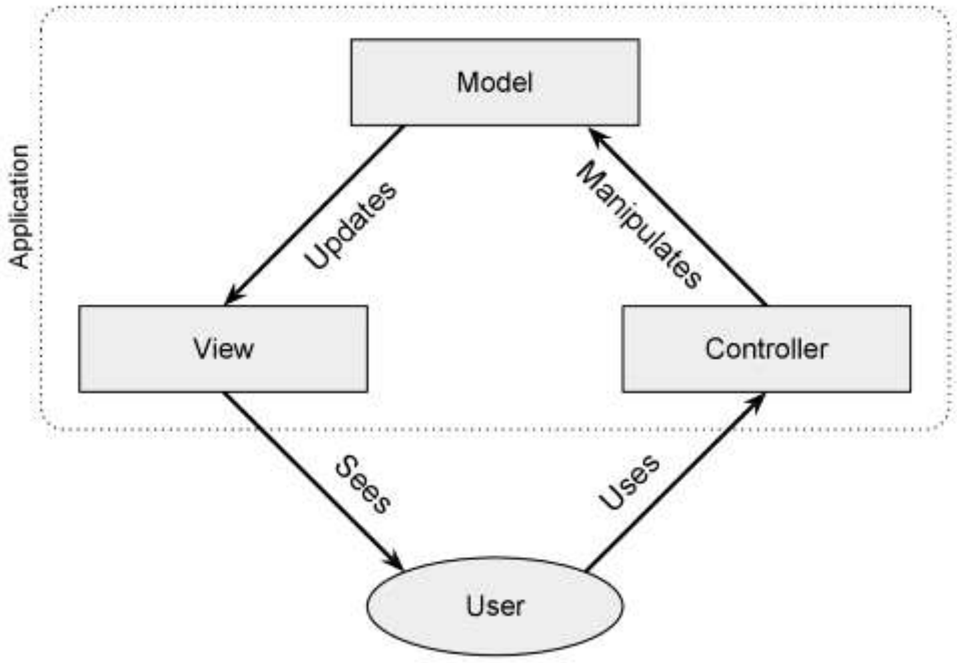


Figure 24 Basic Model-View-Controller pattern with relationship to user

The View manages the graphical output, the Controller interprets user inputs and commands the model to change as appropriate, and the Model manages the behavior and data of the application, responds to requests for information about its state, and responds to instructions to change states. This separation of responsibilities is necessary to ensure scalability as well as stability in graphical user interfaces.

Because the ICEF is a real-time application with remote data sharing capabilities, the Model-View-Controller pattern is used in the client application where the line between the user interface and the pure data is drawn. There are two loops in the ICEF architecture (Figure 25 and Figure 26), since the software has both 2D and 3D interfaces.

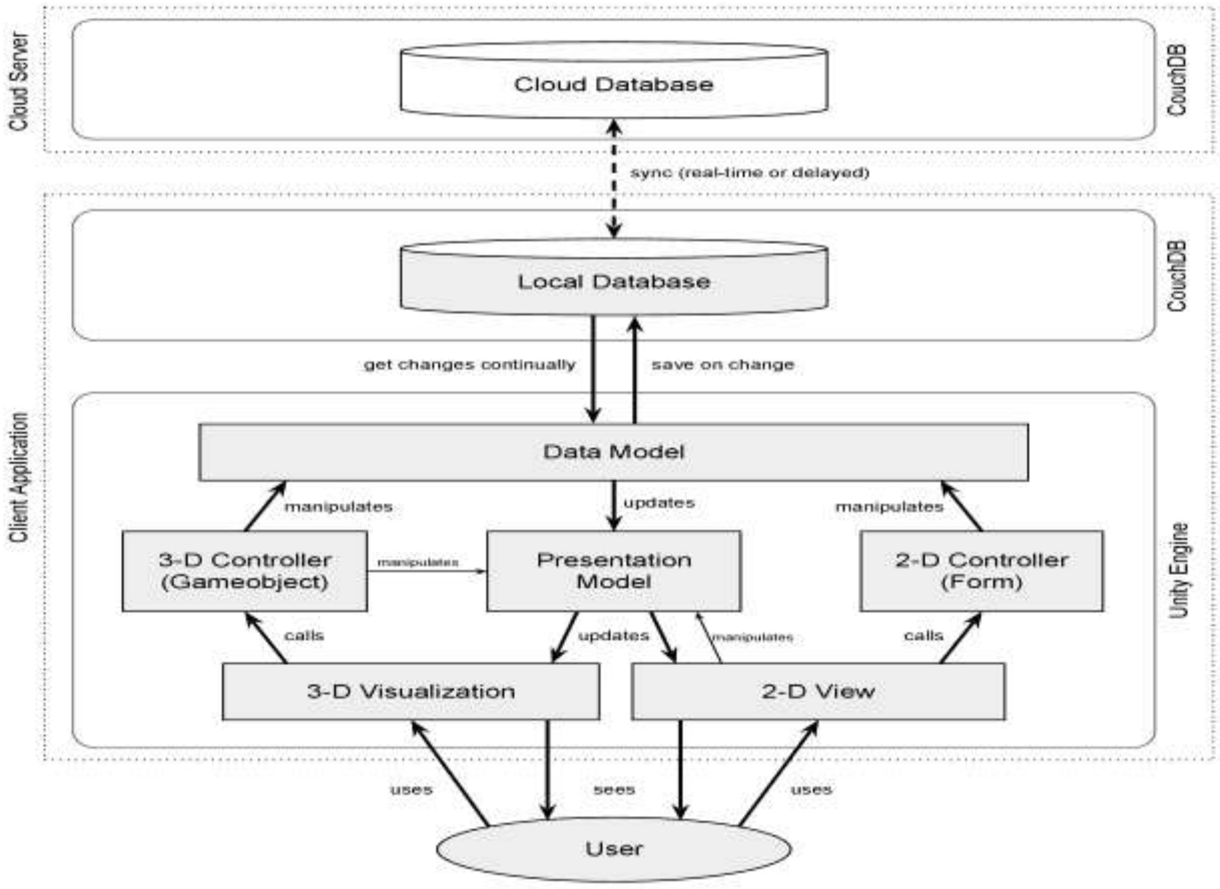


Figure 25 ICEF Architecture

The ICEF logical internal architecture is shown below.

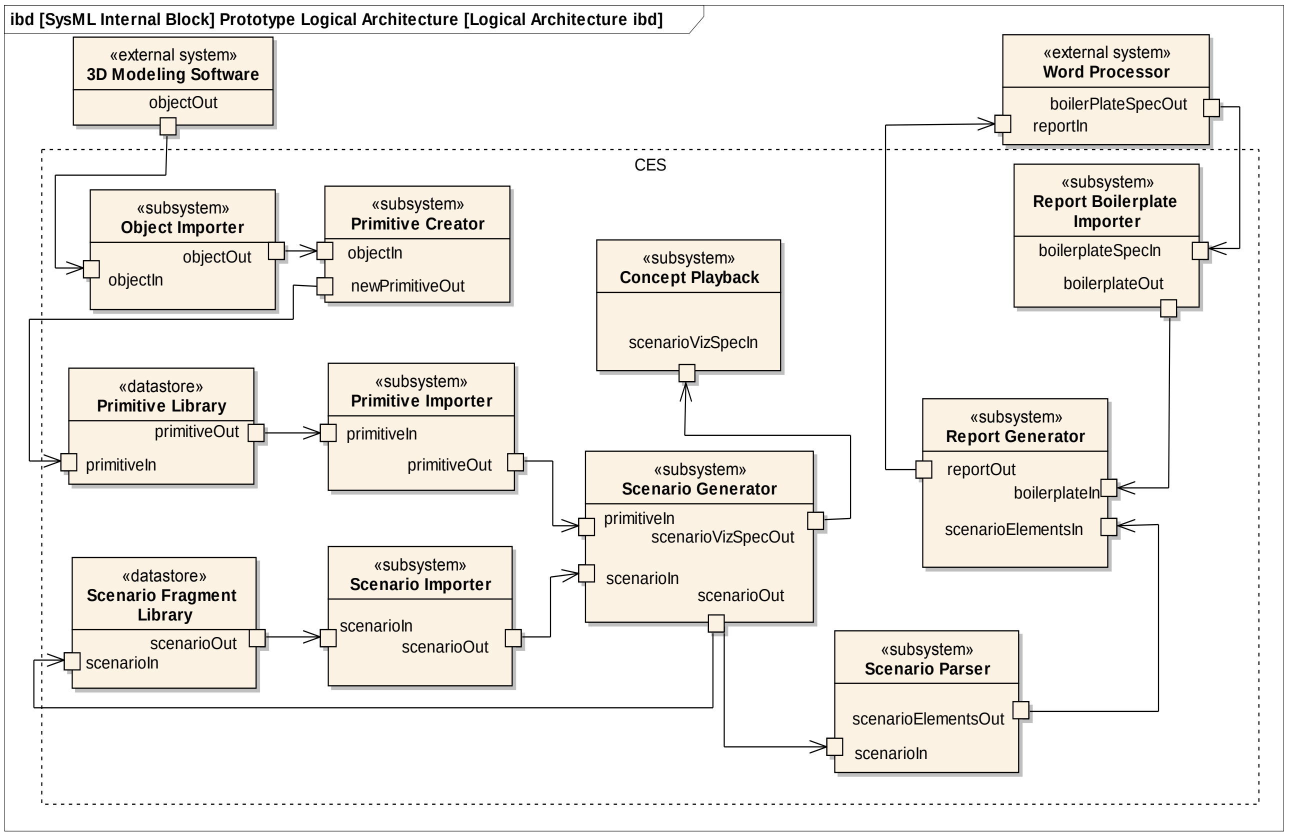

ibd [SysML Internal Block] Prototype Logical Architecture [Logical Architecture ibd]
«external system» 3D Modeling Software
objectOut
CES
«subsystem» Object Importer
objectOut
objectIn
«subsystem» Primitive Creator
objectIn
newPrimitiveOut
«subsystem» Concept Playback
scenarioVizSpecIn
«external system» Word Processor
boilerPlateSpecOut
reportIn
«subsystem» Report Boilerplate Importer
boilerplateSpecIn
boilerplateOut
«datastore» Primitive Library
primitiveOut
primitiveIn
«subsystem» Primitive Importer
primitiveIn
primitiveOut
«subsystem» Scenario Generator
primitiveIn
scenarioVizSpecOut
scenarioIn
scenarioOut
«subsystem» Report Generator
reportOut
boilerplateIn
scenarioElementsIn
«datastore» Scenario Fragment Library
scenarioOut
scenarioIn
«subsystem» Scenario Importer
scenarioIn
scenarioOut
«subsystem» Scenario Parser
scenarioElementsOut
scenarioIn

Figure 26 ICEF Internal Architecture

5.1 Data and Presentation Models

Within the ICEF environment, there are two different models – one to handle the domain data that is being stored and shared between users (the Data model) and one to handle the execution of a specific application (the Presentation model). The Data model has classes, shown in Table 3, which relate to the storyboarding concepts discussed in Section 5.2 Terminology of ICEF & Storyboarding Techniques.

Table 3: ICEF Data Class Model Listing

Data Model Class	***Description***
Building	The environment of a location, including its 3-D model
Domain	The domain as specified by Actors and Actions
Link	Connects actors of an action (if any)
LinkType	Indicates the type of a link (the verb)
ObjType	Class type of an actor, defining its behavior (e.g. Person, Information, Equipment)
Primitive	The specific class of an actor, defining its Domain, ObjType, and the 3-D model associated with it
PrObject	Represents the Actor, defining its Primitive and membership relationship in any Organizations or Teams
PrObjPos	The position of an Actor during a specific Action
ScAction	An Action, specific to one or more domains, may contain a Link, which defined the Actors involved in the Action
Scenario	Contains an ordered list of Scenes and may have a human-readable summary
Scene	Specifies a location, some Actors, an ordered list of Actions, and may have a human-readable description
ScnrTalk	The conversation shared between Scenario authors

During user workshops, the need to add additional Actors or Actions beyond those available in the existing database. This feature was added by creating an additional field and allowing the user to insert a placeholder model for objects, actors or actions which do not have an available 3D model (for Actors) or activity listing (for Actions).

The relationship between the data model classes is shown below in Figure 27.

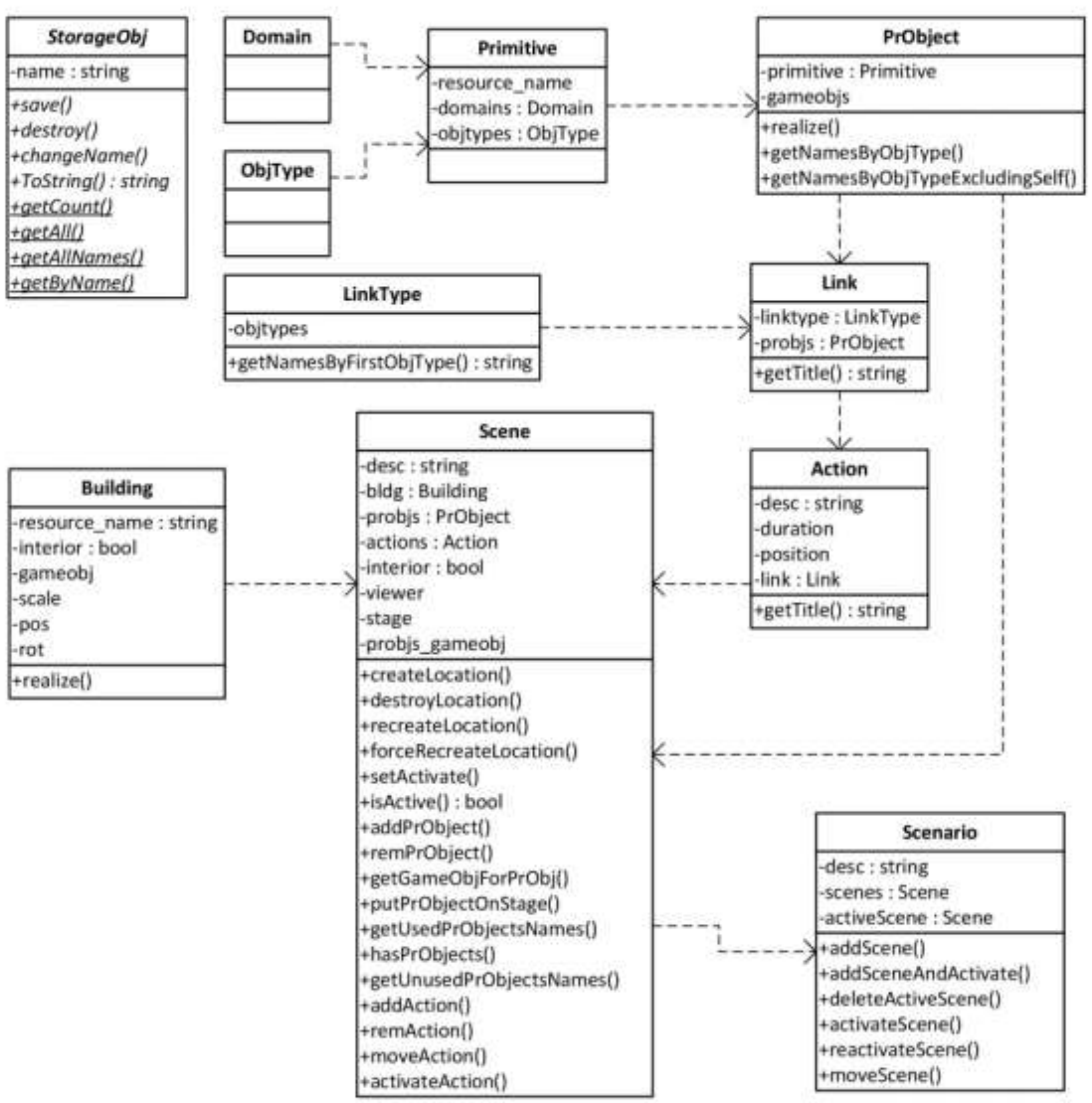


Figure 27 Data Model Class Relationships

5.2 TERMINOLOGY OF ICEF & STORYBOARDING TECHNIQUES

The idea of storyboarding comes naturally to the discussion of CONOPS, and the application takes the unyielding and relentless march of linear time and adapts to it. As a scenario unfolds, the viewer can see it develop in a natural way. However, when authoring a scenario, greater flexibility is needed – there may be many modeling iterations and many different configurations tested before a suitable CONOPS can be modeled and then generated. This requires that the author be able to view a graphical representation of the scenes which comprise the scenario, as well as their sequence and the order of actions within each separate scene. Some of the frequently used terms are shown below.

Scenario - a set of activities comprising the use case of a system, containing the functional flow from the system user perspective

Scene – an ordered set of actors and actions, in a specified location.

Location – the geographical site where actors congregate and actions occur. Each scene is bounded to one location, but a single location may appear in many scenes.

Actor – the basic elements of a system which are involved in actions. An actor may be a person, a device, information, etc.

Team & Organization – groupings of basic elements, and a new entity which can participate in actions. Actors have a membership relationship with teams and organizations.

Actions – the building blocks of scene flows, they describe hour activities are executed in a scene. Actions are a partially ordered set, and can be projected onto a one-dimensional list. Concurrency of actions is possible.

Duration – the length of time an action pertains. The duration of a scene is the minimum time required to complete all the contained actions.

5.3 Exporting CONOPS to SysML - Tool Interoperability

An important part of building any engineering tool is interoperability with other engineering tools used during the development lifecycle. In fact, one of the major challenges laid forth by the INCOSE Model-Based Systems Engineering initiative has been interoperability of model-based systems engineering tool across the system development lifecycle. Since ICEF is among the first tools of its kind, attempting to extend the principles of MBSE to the earliest parts of the system engineering lifecycle, emphasis was placed on enabling interoperability with industry standard SysML tools.

Looking at natural language equivalents of the ICEF terminology, a link can be extended to matching SysML entities and the diagrams for which they are applicable. This relationship can be seen in Table 4. This table makes use of the following abbreviations for specific SysML diagrams:

Structural Diagrams

- bdd = block definition diagram
- ibd = internal block diagram

Behavioral Diagrams

- uc = use case diagram
- act = activity diagram

- par = parametrics diagram

Table 4: Translating natural language to ICEF to SysML

Natural Language	ICEF Component	SysML Entity (diagram)
Noun	Object (system, system components)	Block (bdd, ibd)
Person	Actor (user, stakeholder, person)	Actor (bdd, uc)
Verb	Actions (user/system action & reaction)	Activity/Use Case (act/uc)
Property	Attribute (performance parameter)	Property (bdd, par)
Team	Team (groups, organizations)	Aggregation Relationship (bdd)

This mapping of ICEF components allowed for translation between ICEF scenarios and SysML entities. Additionally, given the storyboarding structure of ICEF scenarios, the user-entry field for scenario and scene were used to define high level use cases in the uc diagram.

With this mapping in place, developers implemented ICEF capability for SysML output. At any time during the modeling session, a user can click the button labeled "Export SysML for Scene" and ICEF will generate SysML diagrams.

Behind the scenes, three C# files were written to query the CouchDB database's contents and extract all relevant information for the three SysML diagram types (ActivityStorage.cs, BlockStorage.cs and UseCaseStorage.cs). Another set of C# scripts provide the definition model classes to generate SysML (Activity.cs, Block.cs and UseCaseStorage.cs), while the XMIGeneration() function is used to parse and translate the data, and generate a SysML1.1 standard XMI file.

The user can then open the SysML tool Enterprise Architect, import the XMI files, and SysML bdd, uc and act diagrams will be displayed. A challenge in this capability is the variance of XML schema required by specific SysML tools. The XMI file produced by ICEF matches the schema required by Enterprise Architect, which was chosen due to its simplicity. However, a number of other commercial SysML tools provide modules that can convert Enterprise Architect models to other, more complex XML schemas. For example, an XMI file generated from an ICEF scenario was successfully imported to Enterprise Architect, and using the NoMagic Enterprise Architect converter, the resulting diagrams were imported to MagicDraw. However, some revision and rework is needed to "fix" the translated diagrams with MagicDraw, especially the act diagram.

With this capability, users who model operational scenarios within ICEF can export contents of the scenes directly to a SysML tool. Although the functionality has not been integrated into ICEF yet, this mapping of ICEF components to SysML diagrams would also allow for translation of SysML diagrams to ICEF models.

5.4 INTERFACING WITH MATLAB

Within the RT 031 effort, the research sponsor requested a proof of concept demonstration of ICEF execution using MatLab as the underlying simulation engine. The proposed use case underpinning this demonstration was an analysis of motion profile of military vehicles. Data files were created containing properties such as personnel and fuel capacity, mean and max speed, and acceleration of a variety of Army vehicles including the Humvee, JLTV, MRAP, M113 APC and Stryker platforms. Upon entering the ICEF environment, a C# script prompts MatLab to open in the background and load a prewritten motion script. The user is asked to select a vehicle's distance of travel, preferred speed and acceleration. The acceptable selection ranges are context specific, dependent on the user's choice of vehicle. When the user hits the play button, the vehicle's properties, along with the user specified parameters, are transmitted to MatLab using TCP/IP link. In real time, MatLab determines the motion profile of the vehicle, and feeds the information back to ICEF. With less than a second delay, ICEF uses the input from MatLab to move the vehicle down the road, constantly updating the user with the time, distance, velocity and acceleration. Once the designated distance has been traveled, MatLab produces a full report of the vehicle's movement, which can be exported to Excel and other software tools for further analysis

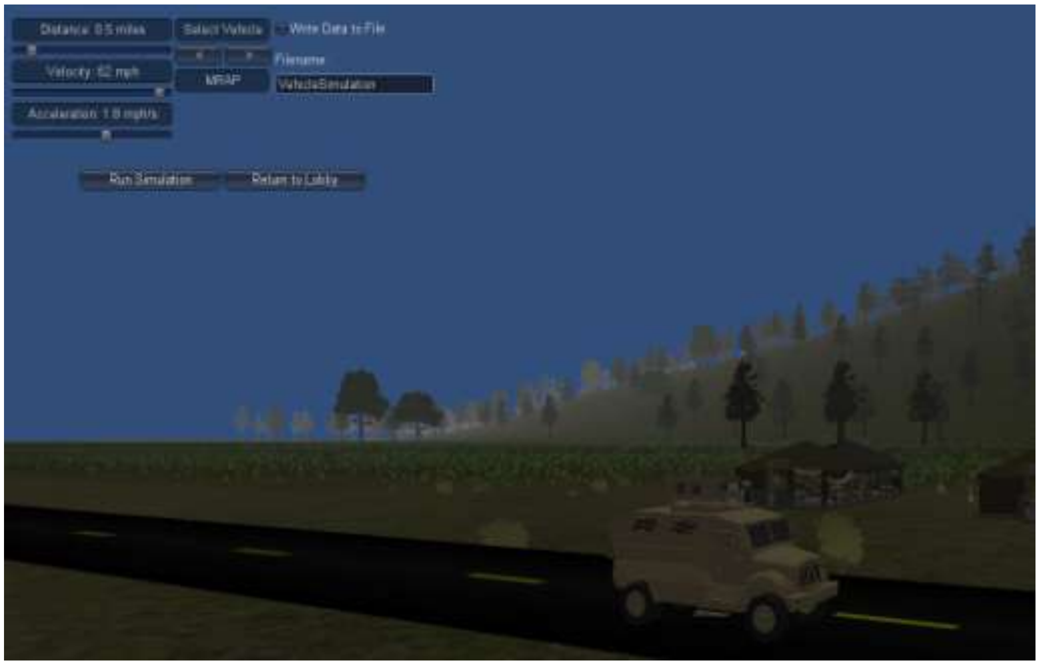

Distance: 0.5 miles
Velocity: 62 mph
Acceleration: 1.8 mph/s
MRAP
Filename
VehicleSimulation

Figure 28 ICEF MatLab Interface

Additional details regarding the ICEF/MatLab interface can be seen in Appendix F: Unity_Code_MATLAB_Invocation_Used_in_RT31 and Appendix G: MATLAB script Vehicle Simulation.

6.0 Scenario Building

Starting the latest version of ICEF software, the user is presented with a four panel menu as shown in Figure 29.

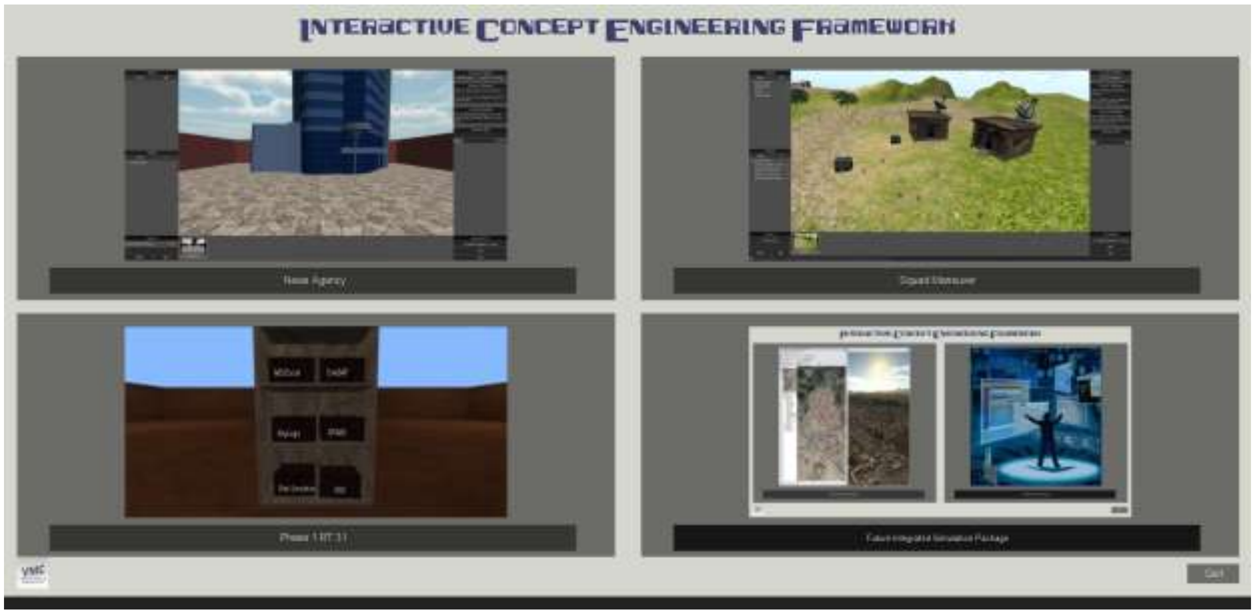
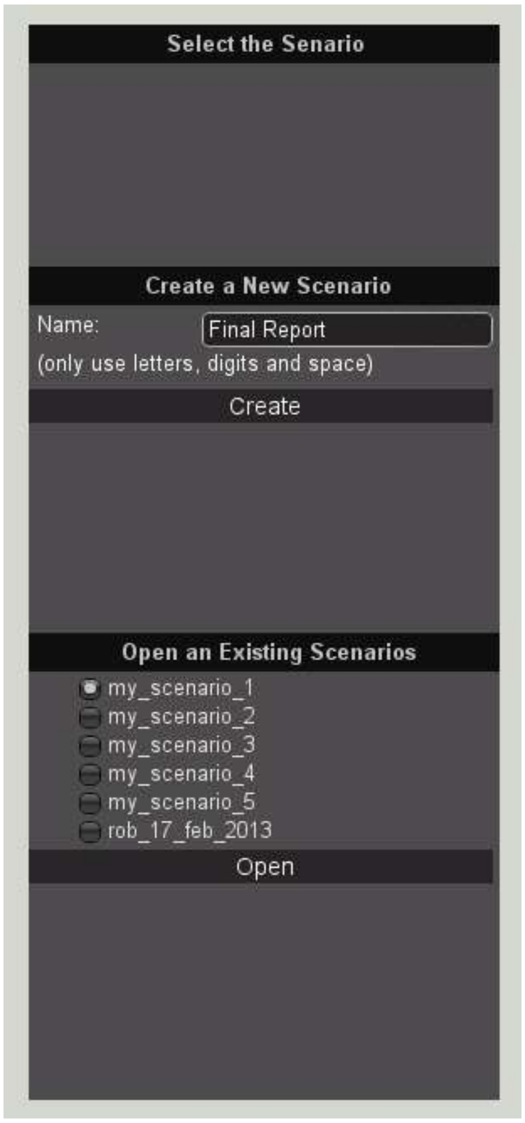

INTERACTIVE CONCEPT ENGINEERING FRAMEWORK
News Agency
Squad Maneuver
Phase 1 RT 31
Future Integrated Simulation Package
Quit

Figure 29 Current Author Start Menu

While the code for RT30/30a/31/31a has been integrated into one package, utilizing a single shared infrastructure and common capabilities, each button within this menu will direct the ICEF users to an environment tailored towards a specific domain.

The News Agency domain (upper left) focuses on RT30/30a scenarios. The Squad Maneuver domain (upper right) was custom built based on RT31a scenarios. The Phase 1 RT 31 option (lower left) presents a standalone demonstration of ICEF interface capabilities. While the most recent scenario building capabilities focus on the CONOPS Author as the primary user, the Future Integrated Simulation option (lower right) acts as a placeholder for future development of a CONOPS Developer-specific environment. This area would provide a more technical user with access to advanced capabilities, such as direct CouchDB and Unity development environment access, to work across all domains and capabilities of ICEF to assist CONOPS authors and analysts carry out their modeling and simulation activities.

Entering the domain of choice, a new dialog box opens requiring the Author to provide a new project name, or select an existing scenario from the database. (Figure 30).



Select the Senario

Create a New Scenario

Name: Final Report

(only use letters, digits and space)

Create

Open an Existing Scenarios

- [✓ my_scenario_1]
- my_scenario_2
- my_scenario_3
- my_scenario_4
- my_scenario_5
- rob_17_feb_2013

Open

Figure 30 Creating a new Scenario

Once the scenario is created in the database, the software will determine a default initial location. In the case of the News Agency domain, the default location is outside the news agency (Figure 31)

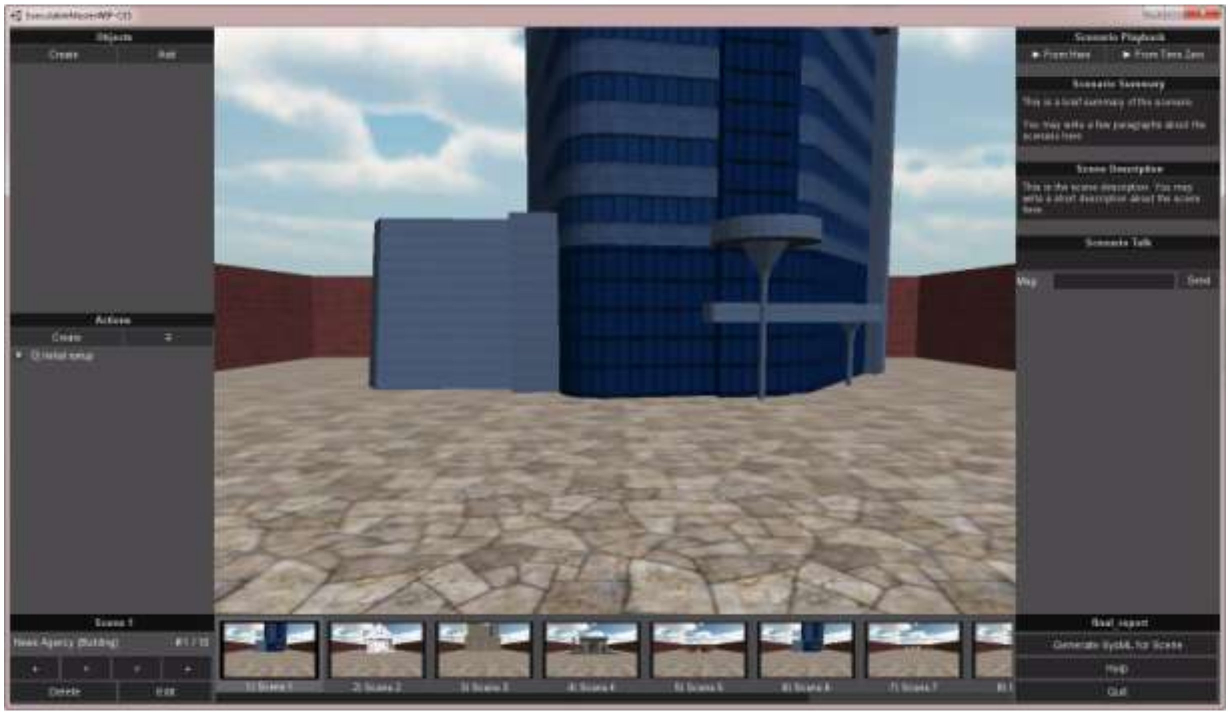
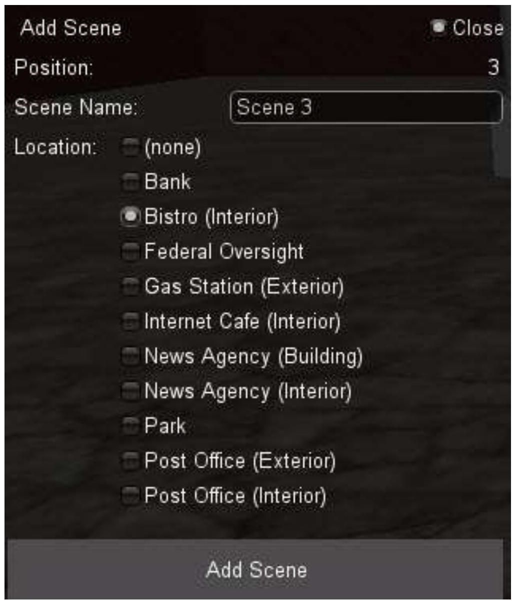


Figure 31 Outside the News Agency

6.1 NEWS AGENCY SCENARIO

Scenario development for RT30a is carried out by creating a number of scenes in which the use cases will occur. A scenario is then built up as a collection of multiple scenes. Scenes can take place at the same location, however, the RT30a ICEF also allows the user to specify an entirely different location. To add a new scene, the plus (+) sign in the lower left section is press, and doing so opens the following menu (Figure 32). This menu shows the locations in the current domain database.



Add Scene ● Close

Position: 3

Scene Name: Scene 3

Location:
- (none)
- Bank
- [✓ Bistro (Interior)]
- Federal Oversight
- Gas Station (Exterior)
- Internet Cafe (Interior)
- News Agency (Building)
- News Agency (Interior)
- Park
- Post Office (Exterior)
- Post Office (Interior)

Add Scene

Figure 32 Adding a Scene Dialogue Box

The following figures (Figure 33, Figure 34, Figure 35, Figure 36, Figure 37, Figure 38, and Figure 39) show the rest of the locations available in the New Agency Scenario. As a note, high quality graphics was not the goal. The software was being developed to test the research question. This is not meant to be production quality, but instead, representative of what production software might be capable of performing.

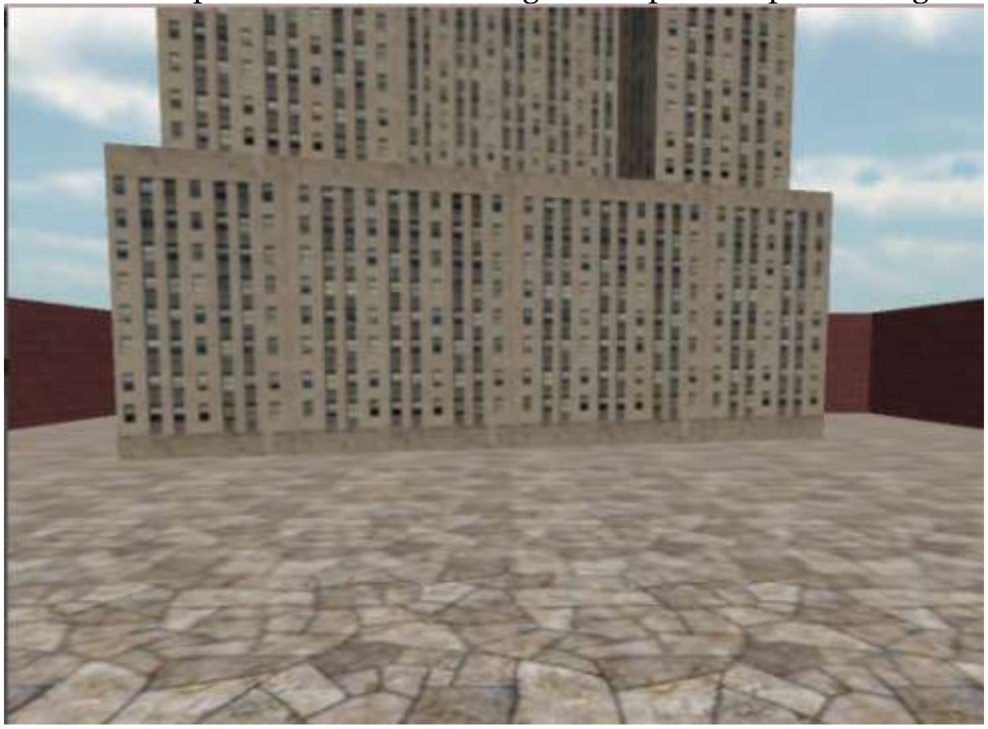


Figure 33 Federal Oversight Building

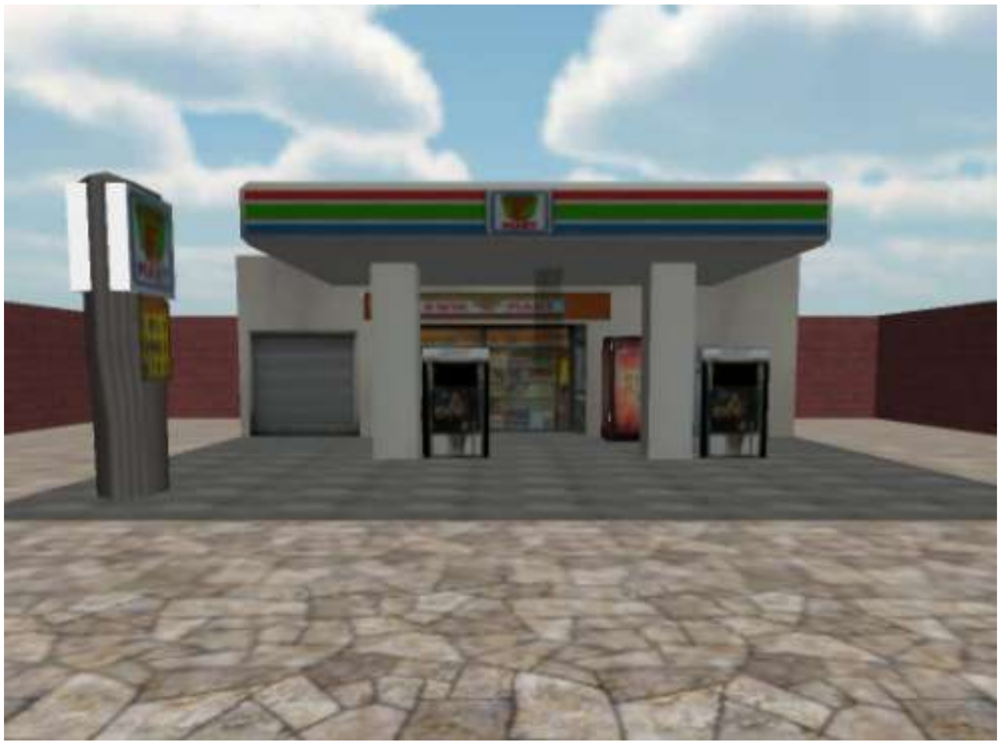


Figure 34 Gas Station

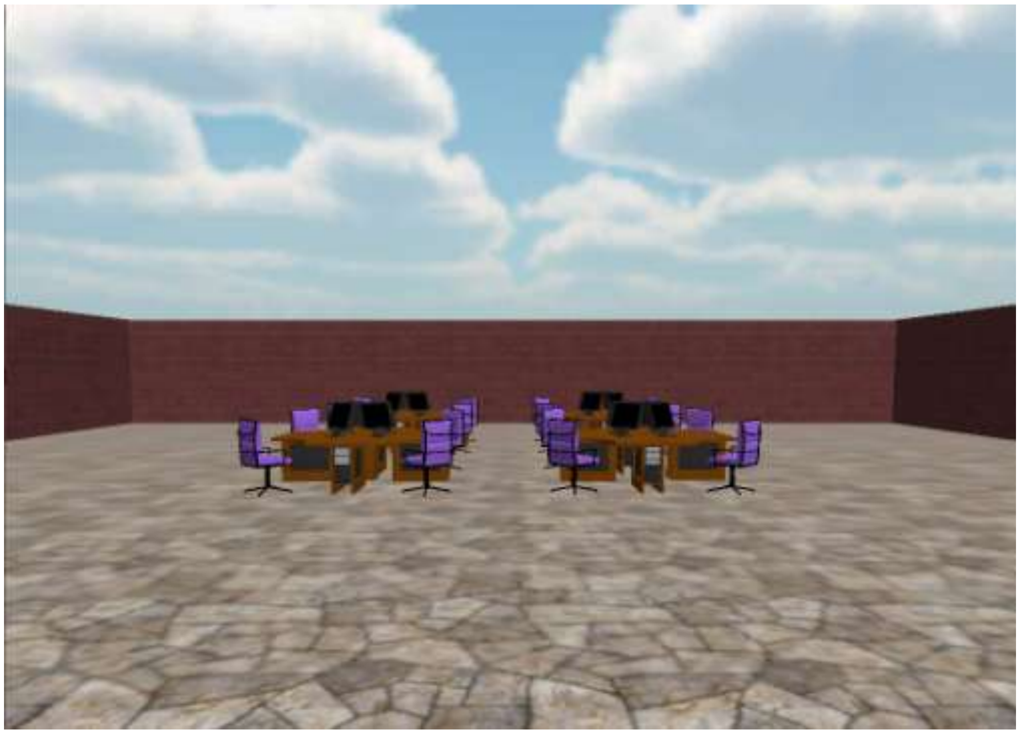


Figure 35 Internet Cafe

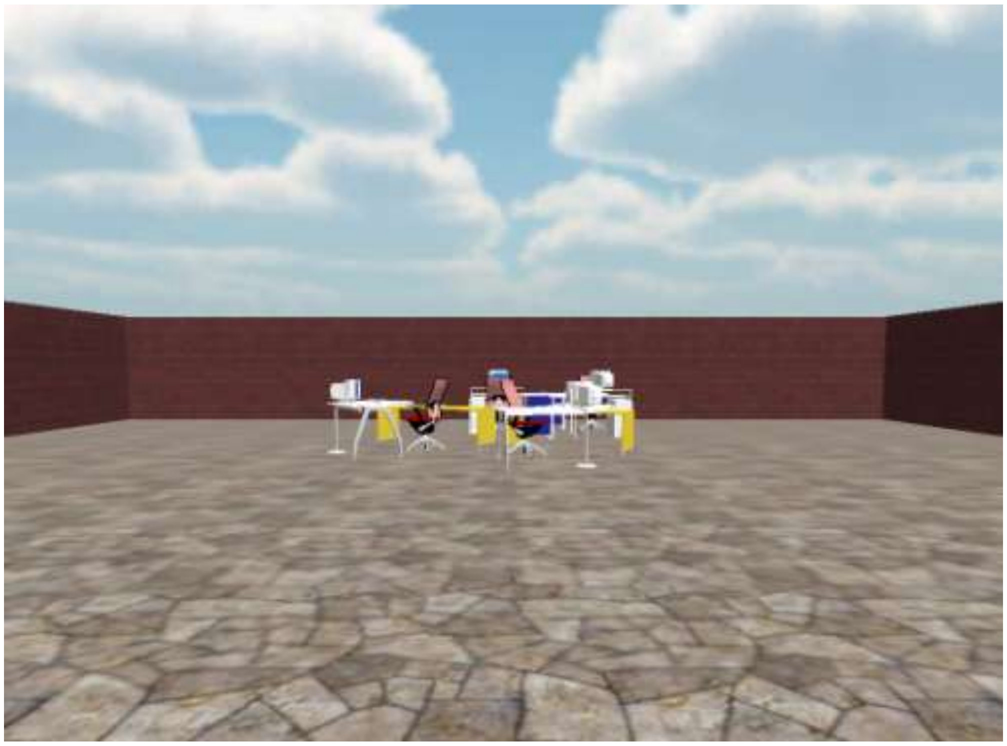


Figure 36 Workspace inside the News Agency

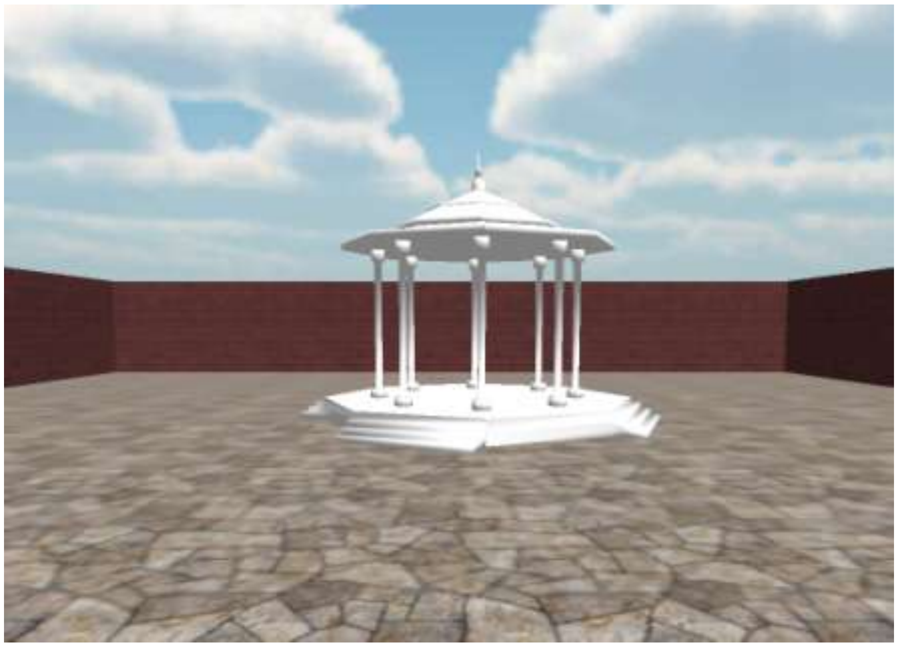


Figure 37 Park

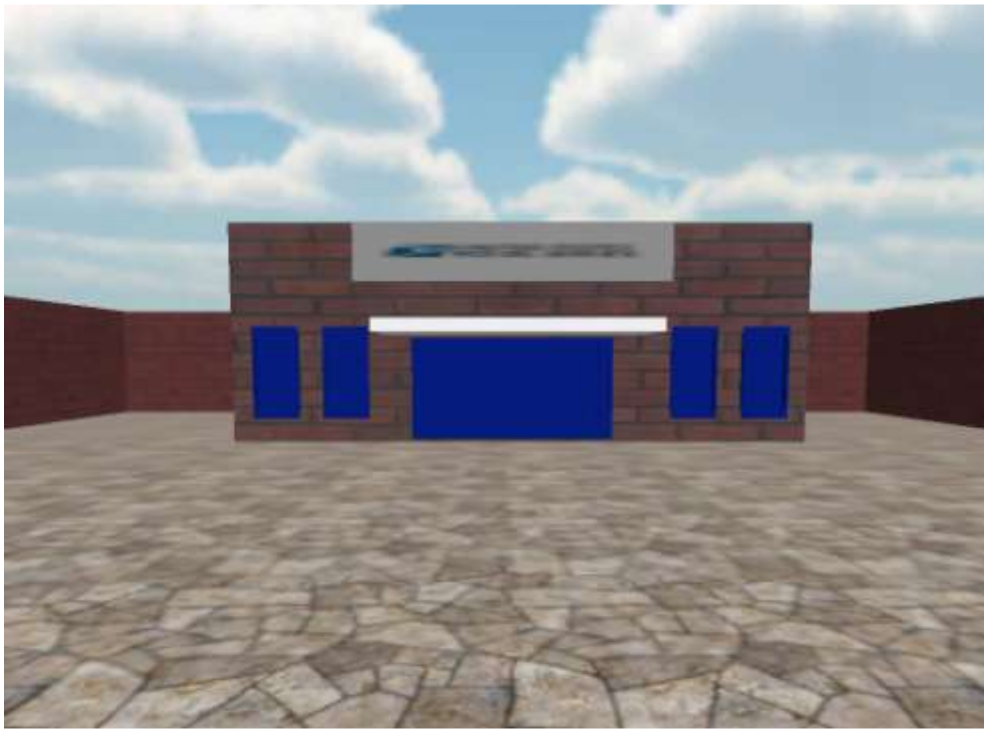


Figure 38 Post Office exterior

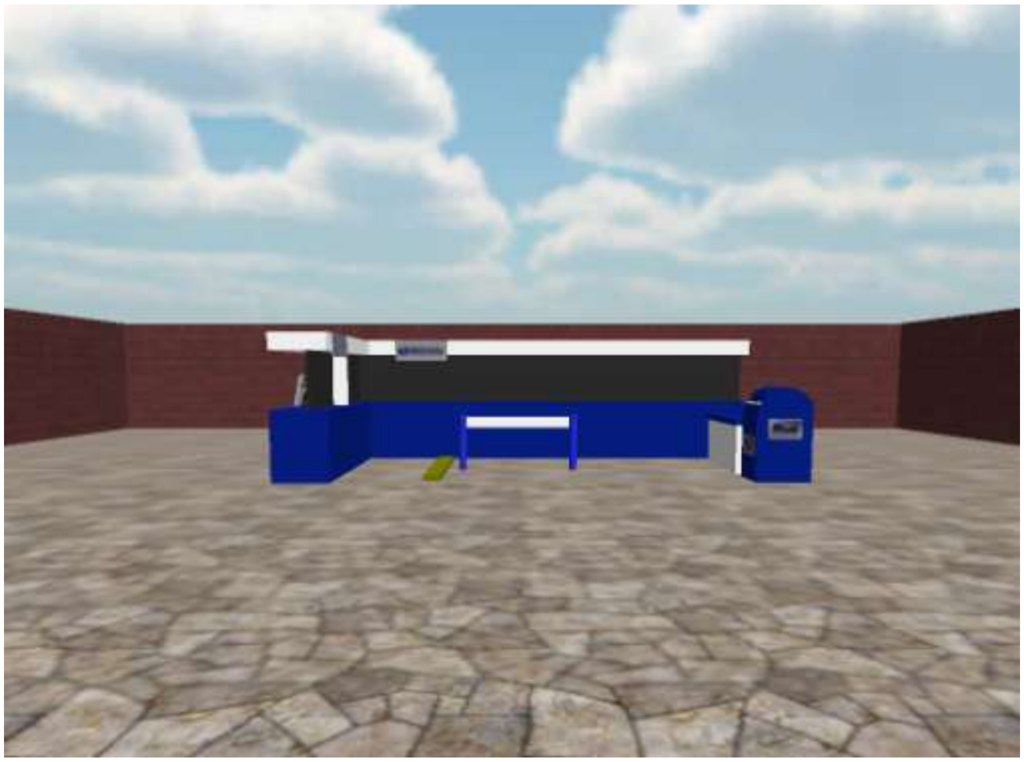


Figure 39 Post office interior

6.2 Squad Maneuver Scenario

In RT-31a the use case examined was a tactical contact-to-fire scenario, in which the locale remained static and the movement of the camera determined which specific section of the locale served as the "room" of a scene (Figure 40). This approach capitalizes on the already-existing scene construction model.

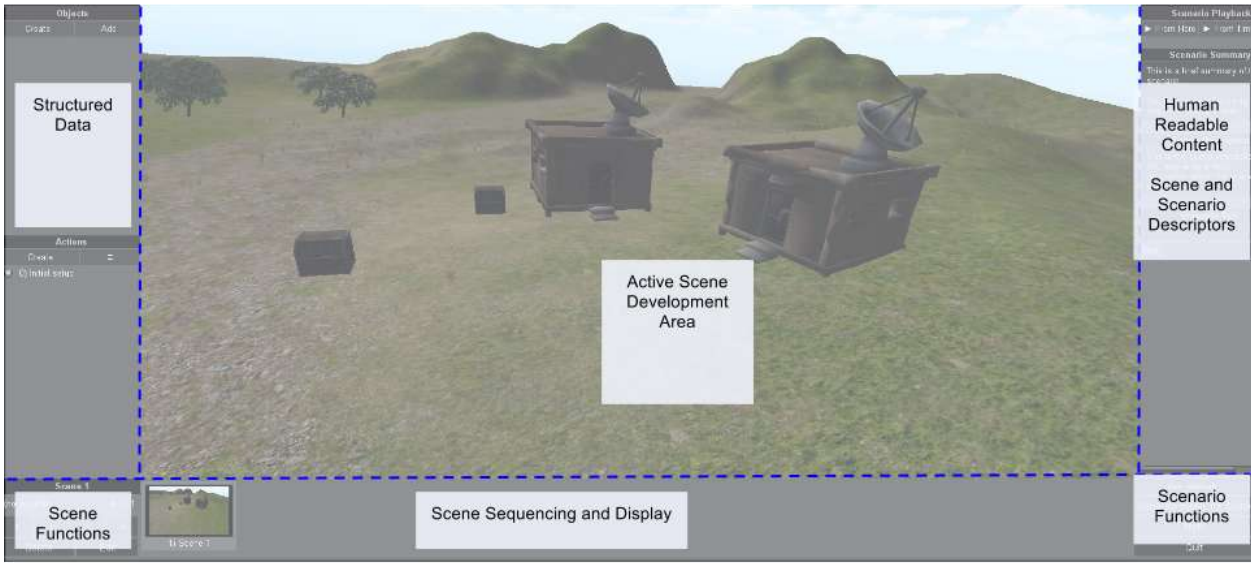

Structured Data
Human Readable Content
Scene and Scenario Descriptors
Active Scene Development Area
Scene Functions
Scene Sequencing and Display
Scenario Functions

Figure 40 CONOPS Authoring Interface – User Elements

Within the scenario, the user can create, specify, add, and modify objects specific to the contact-to-fire scenario. Once the storyboard is complete, the user can do a playback from time zero and watch as the simulation unfolds. The current release of the prototype internalizes the health and ammunition measures, using a random number generator for health (if a soldier is hit with fire) and a uniform distribution for firing rates to determine ammunition remaining.

An example scenario building sequence and the environment representations are shown below. The scenario is simplified to highlight the functionality of the software and its flexibility.

Initial Scene (Figure 41): The application provides the user with an uneven terrain, with some reinforced buildings, along with outbuildings and some flora. The buildings are 3D models, and the author can use the keyboard and mouse to circle and examine them.

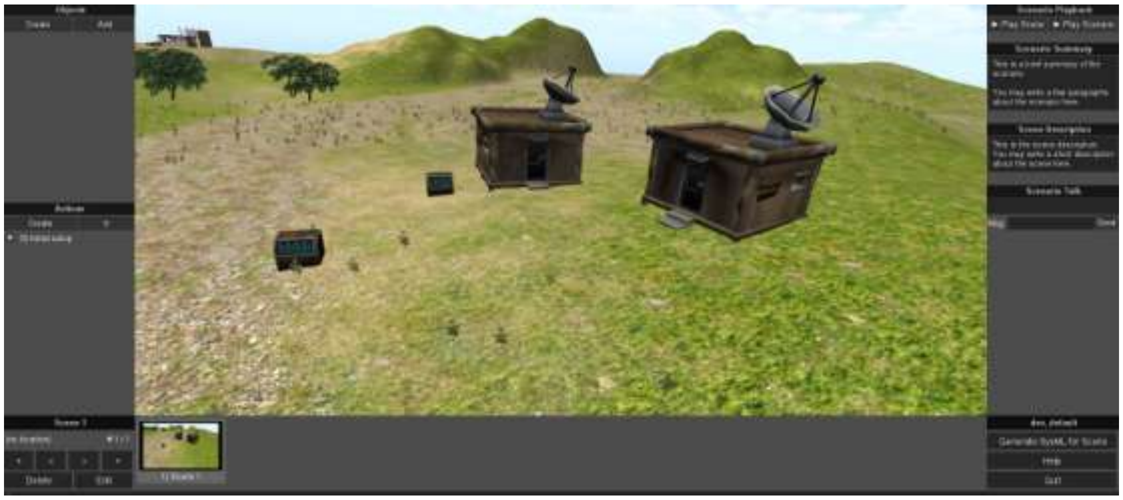


Figure 41 Initial Scene = Battlefield domain

Once the scene is opened, the author can add personnel, both friendly and enemy soldiers. The menu for object addition also contains some placeholders as well as the abstract class of "Team." (Figure 42) The placeholders serve as exactly that – objects for which there is no 3D representation available in the domain yet, but for which the author needs a presence in the scenario. The resulting SysML will show the placeholder – and this can serve as an alert to the domain programmer that an additional 3D model is needed. The abstract class of team can have a fluid definition. A team can be a squad, a battalion or, in this case, two soldiers. What is powerful about this representation is that actions can be assigned to the team and all members will perform them – crouch, crawl, walk, run, stand, etc.

While the process of adding personnel did not change, it was expanded for this new use case and domain. In this case, new personnel are also instantiated with 100% health and 100% ammunition ratings (Figure 42). As the soldiers fire and/or are shot, their health may or may not be reduced and their ammunition remaining will be reduced.

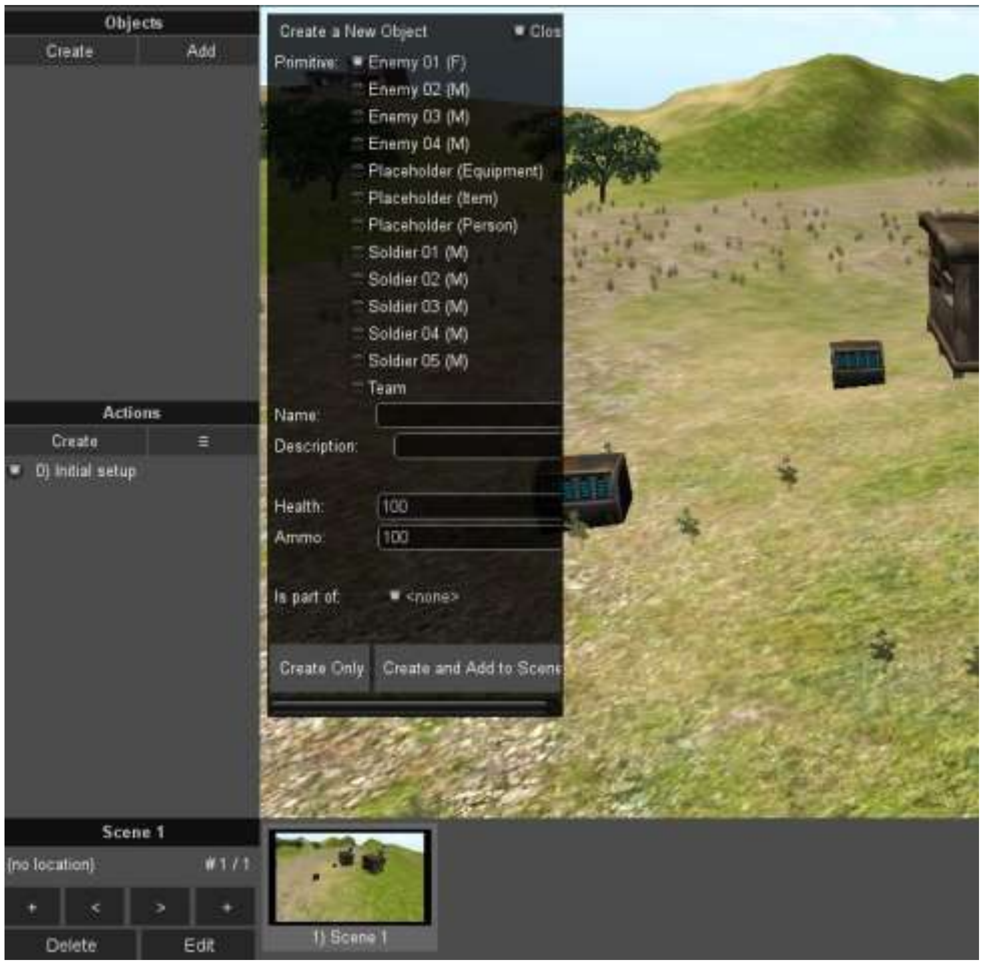


Figure 42 Creation of personnel

The author can easily change the camera angles, as well as pan, zoom in, zoom out, rotate and view all of the surrounding area. This scenario includes two enemy soldiers and two friendly soldiers for this scene. Figure 43 shows the Objects currently residing in the scene. There is also a Team associated with the scene which has no physical representation but exists in the database and can be the subject of any team-related actions.

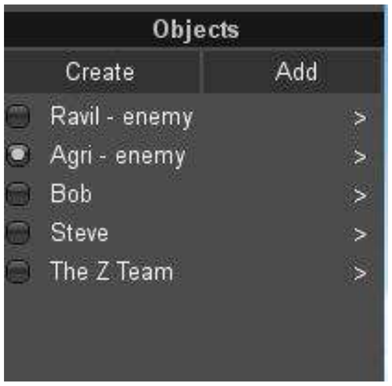


Figure 43 Character listing for scene

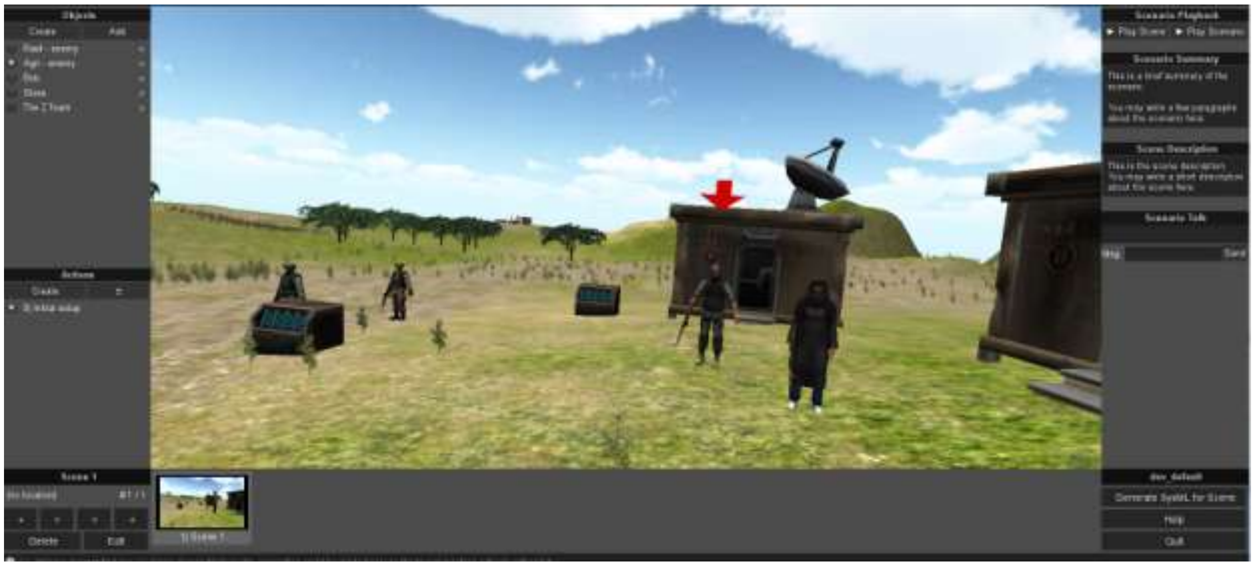


Figure 44 Scene with Soldiers in place

In Figure 44 the Agri-enemy soldier has a red arrow over his head – the red arrow indicates that the character is "selected." The Agri-enemy is selected using a radio button (Figure 45). With the enemy soldier selected, the author can now create an action to be associated with that character.

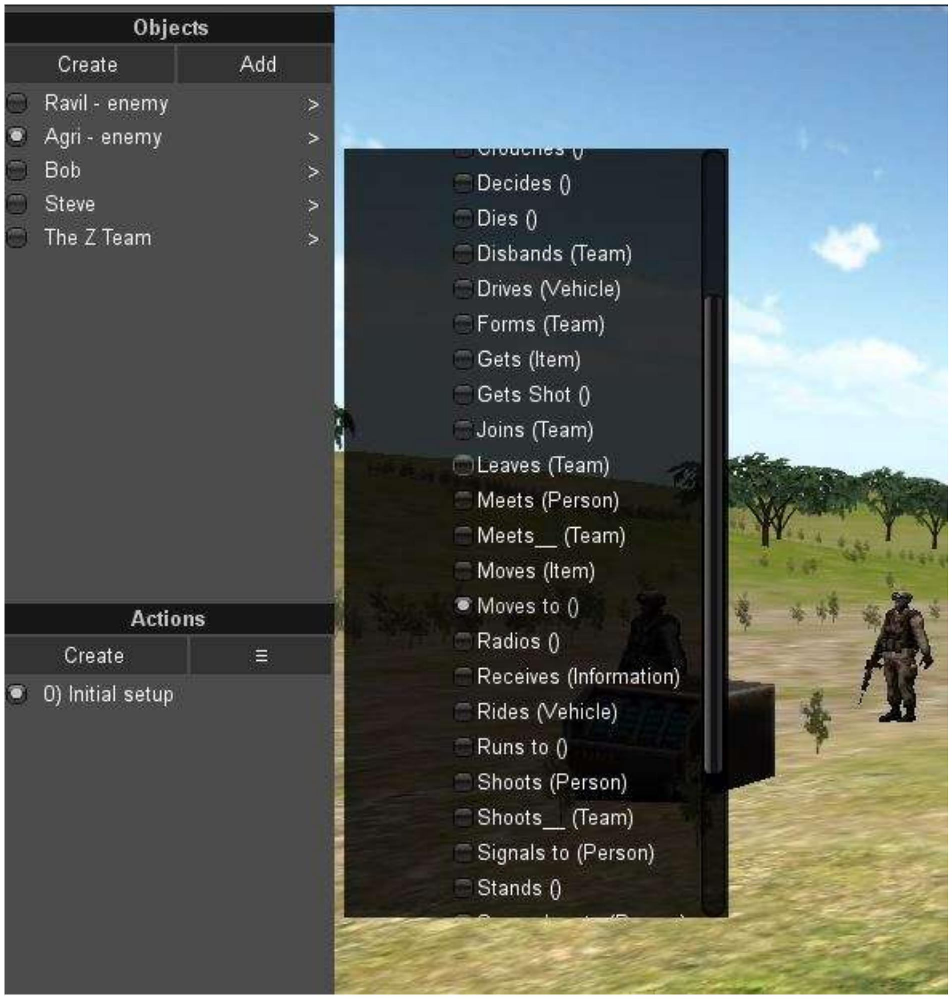


Figure 45 Action Listing for Character

In this example, the action Moves to () for Agri-enemy is selected using a radio button (Figure 45). The verb "moves" is used differently when applied to different characters – there is a "Moves" for an Item. In this case, "Moves to" will assume that the author selects a locale and moves the soldier there. One subtlety which arises when assigning actions is that although the basic "sentence" structure of an activity may be the same, the originators and recipients of that action may be more than simple person-to-person action – it may be the actions of a team toward one person, one person toward a team, or a team toward another team. In order that the display correctly depicts a multitude of possibilities, this construction needed to be considered in the application development.

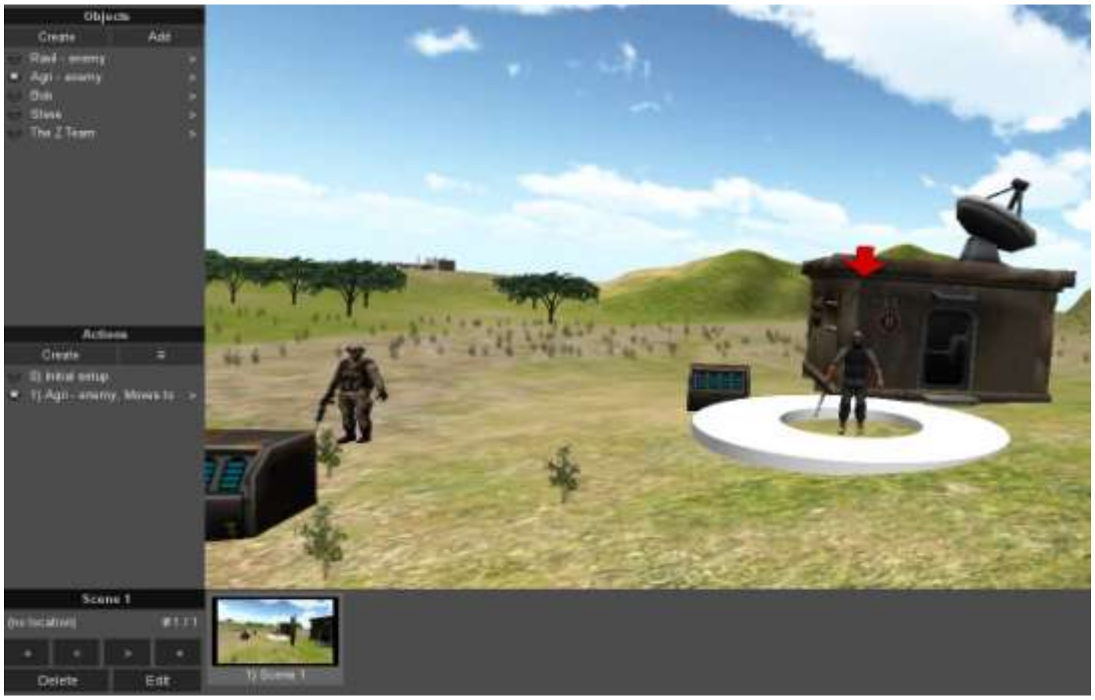


Figure 46 Action with indicated object

The Actor is selected (Figure 46), then moved to the next position. When the simulation is run, the character will stop at this point (Figure 47).

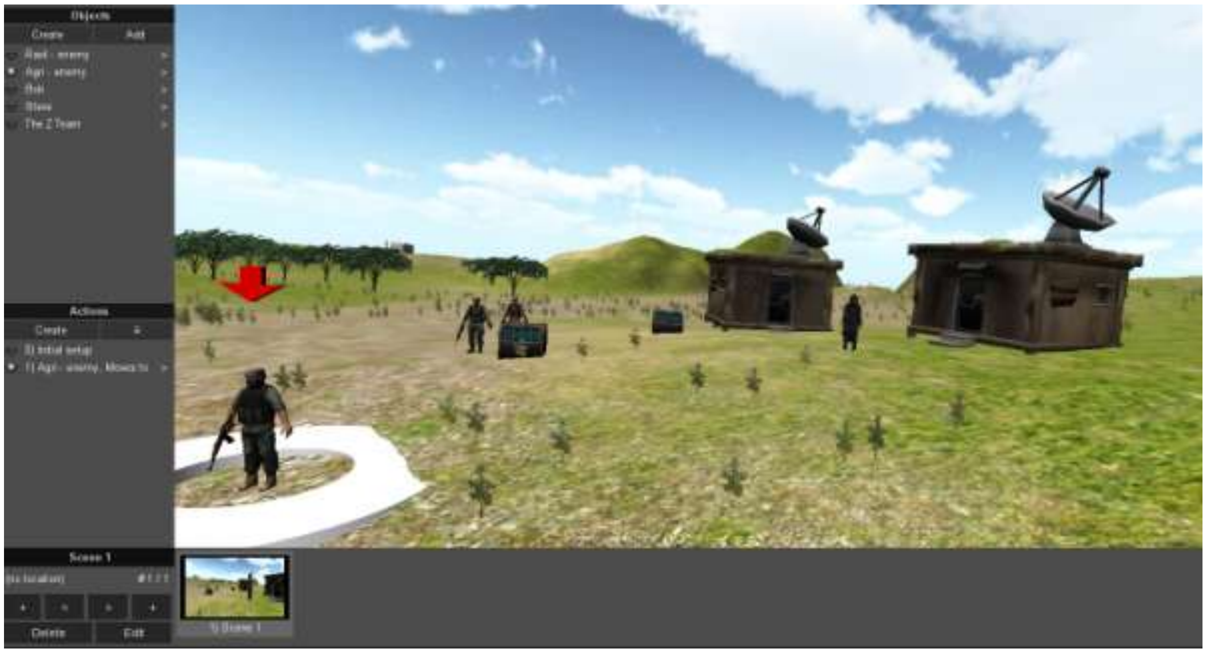


Figure 47 Move Action for Agri-enemy completed

Similarly, the author can indicate that Bob will shoot Agri-enemy, and that Steve will move behind the far bunker (Figure 48).

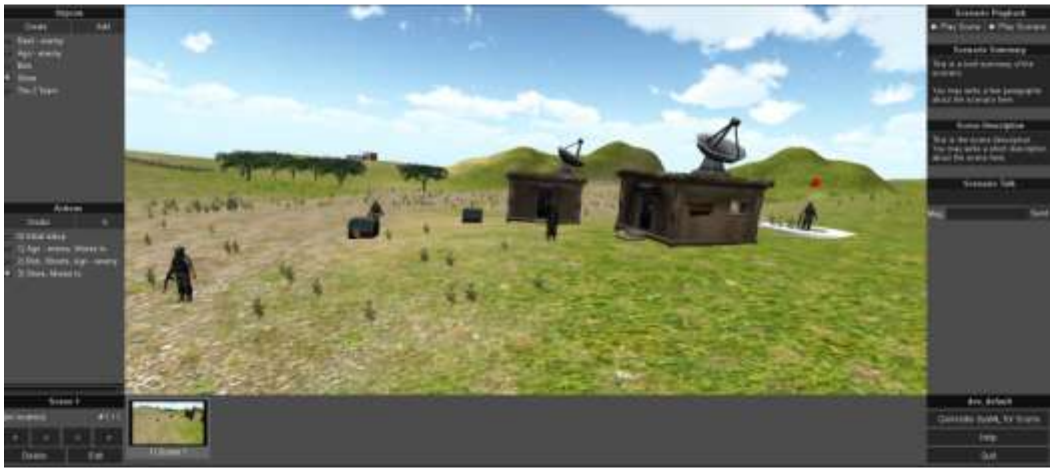


Figure 48 Repositioning of soldier

Finally, commands are given to make Bob move to the far outbuildings, visible in the upper right hand side of Figure 49.

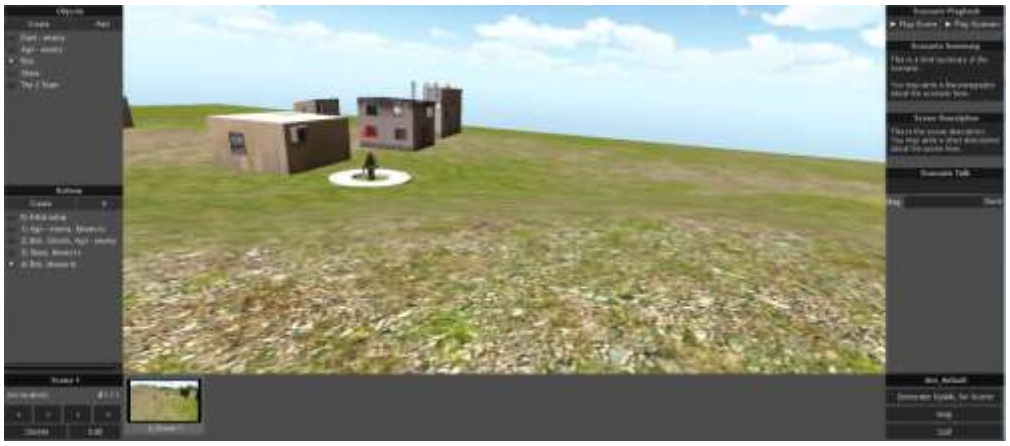


Figure 49 Character moved to far edge of terrain

In addition to the listing of actions which provides a script for the scene, the application is capable of adding scenes which will then continue the action. In this case, if the action has now moved to the outbuildings and will revolve around the character Bob, the author can now add a new scene by pressing the + in the Scene Function area, and a new thumbnail will appear on the bottom Scene Sequencing and Display area (Figure 50).

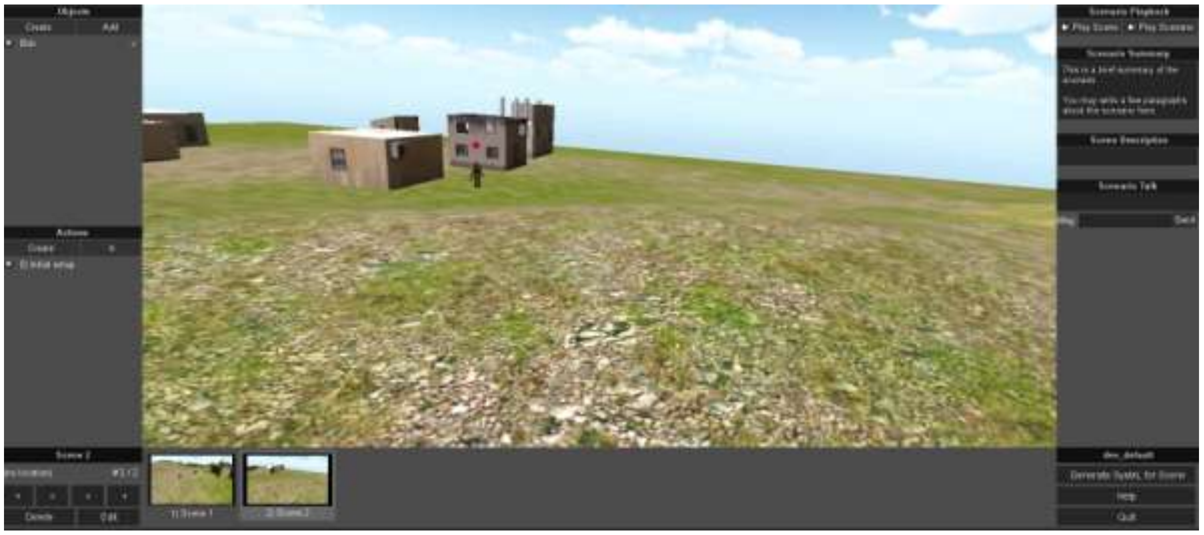


Figure 50 Scene 2 added at the second locale

7.0 WORKSHOP

To determine whether the ICEF development effort is applicable to the problems faced by the sponsor's workforce and can address the established research questions, a workshop was held on December 7, 2011. The workshop participants included: three members of the research team, two of the sponsor's SERC representatives and twelve representatives from the sponsor's organization. The representatives were from a number of areas within the sponsor's organization, in a variety of roles with a wide range of experience. Based on introductions made during the workshop, the following general biographic data was compiled

Table 5: Workshop Participant Data

Position	# Participants	# Years Experience	# Participants
Systems Engineer	11	0-5	2
"Concept Engineer"	5	6-15	3
Software Engineering	4	15-25	1
Tool Development	4	Over 25	7
Manager or Director	3		
System Architect	2		
Requirements Engineer	1		
Lead Systems Engineer	1		
Lead Software Engineer	1		
System Analyst	1		

Throughout the course of the workshop, it became clear that many of the participants were involved in concept development, CONOPS development, and establishing the capability baseline for system development. While there is no position currently called "Concept Engineer", the work that these participants were conducted can be classified as Concept Engineering, therefore the "Concept Engineer" label was added. The workshop participants were briefed on the research being conducted and introduced to the ICEF software. Basic capabilities of the software were described and the participants were asked to use ICEF to model scenarios developed by the research team. The scenarios are described below, followed by selected workshop feedback from participants.

7.1 SCENARIOS

During RT003 Phase 2, the News Agency Scenario was introduced (Cloutier et al., 2010). A taxonomy was developed for creating primitives that would be required to describe News Agency operational scenarios. In RT030, the News Agency taxonomy was utilized to develop a set of specific scenarios to be used for experimenting with and validating the ICEF software. Four scenarios are displayed below using a single sentence and elaborated using a number of statements laying forth the actions required to carry out the scenario. An activity diagram also accompanies each scenario. The activity diagrams were developed as means to graphically represent the scenarios to enhance workshop participant understanding, as well as a tool for iterative design of the scenario.

7.1.1 News Agency Scenario 1

A news agency deploying a reporter and support assets to a new story (Figure 51)

1. An anonymous email is sent to the news agency claiming a major security breach took place in Bank resulting in the exposure of personal and financial information for thousands of customers.
2. NA assigns a reporter to verify the claim by the informant.
3. Reporter goes with crew and talks to the press office of Bank
4. Reporter A feels like he is not being told the truth
5. Reporter A conveys this to Editor, who agrees and sends out a support unit to start collecting footage for the evening edition
6. Support unit meets Reporter A and they record news stories to be played later
7. They transmit their footage to the Editor, who edits the video and prepares it for the evening edition

7.1.2 News Agency Scenario 2

A reporter works to recruit a new contact for a story (Figure 52)

1. Editor needs to have the story confirmed before he can run it
2. Talks to his staff to see who might have contacts to push along the story
3. Reporter Rob says he knows someone at a security firm that handles the Bank's security business
4. Reporter Rob visits his contact, convinces him to share his knowledge and asks him questions
5. The source repeats a similar story to that heard at the Bank, with some inconsistencies.

7.1.3 News Agency Scenario 3

A news agency assigning a reporter to independently corroborate a news source (Figure 53)

1. Before reporting on the story, the Managing Editor needs to corroborate the information gathered thus far.
2. Sends a Staff Reporter to the Fed oversight group

3. Staff Reporter gathers information from information bureau at the Fed oversight group
4. Staff Reporter returns to News Agency and confers with News Editor
5. News Editor evaluates corroboration, decides if it is sufficient for publishing
6. Managing Editor decides whether to run the story or not.

7.1.4 NEWS AGENCY SCENARIO 4

A news agency deploying a new reporter and support assets to follow-up on an existing story (Figure 54)

1. The News Agency receives reactions to a current or previous story.
2. The Editorial Board assesses the potential for a feature story.
3. The Editorial Team assigns Research Team to gather background information/previous stories
4. The Editorial Team assigns a new Reporter and crew to gather new information/interviews
5. A updated news story is created, including information previously collected along with fresh information

7.1.5 PRIMITIVE LIBRARY

Based on the News Agency taxonomy from RT003, along with the workshop scenarios described above, a set of entities or primitives that were required for the workshop was developed. This set includes any objects required to be able to model the News Agency Scenarios described above. Each object also contains the characteristics (attributes) of the object, as well as potential actions/activities (links) it can carry out. The object model for the initial ICEF Primitive Library can be seen in Figure 55.

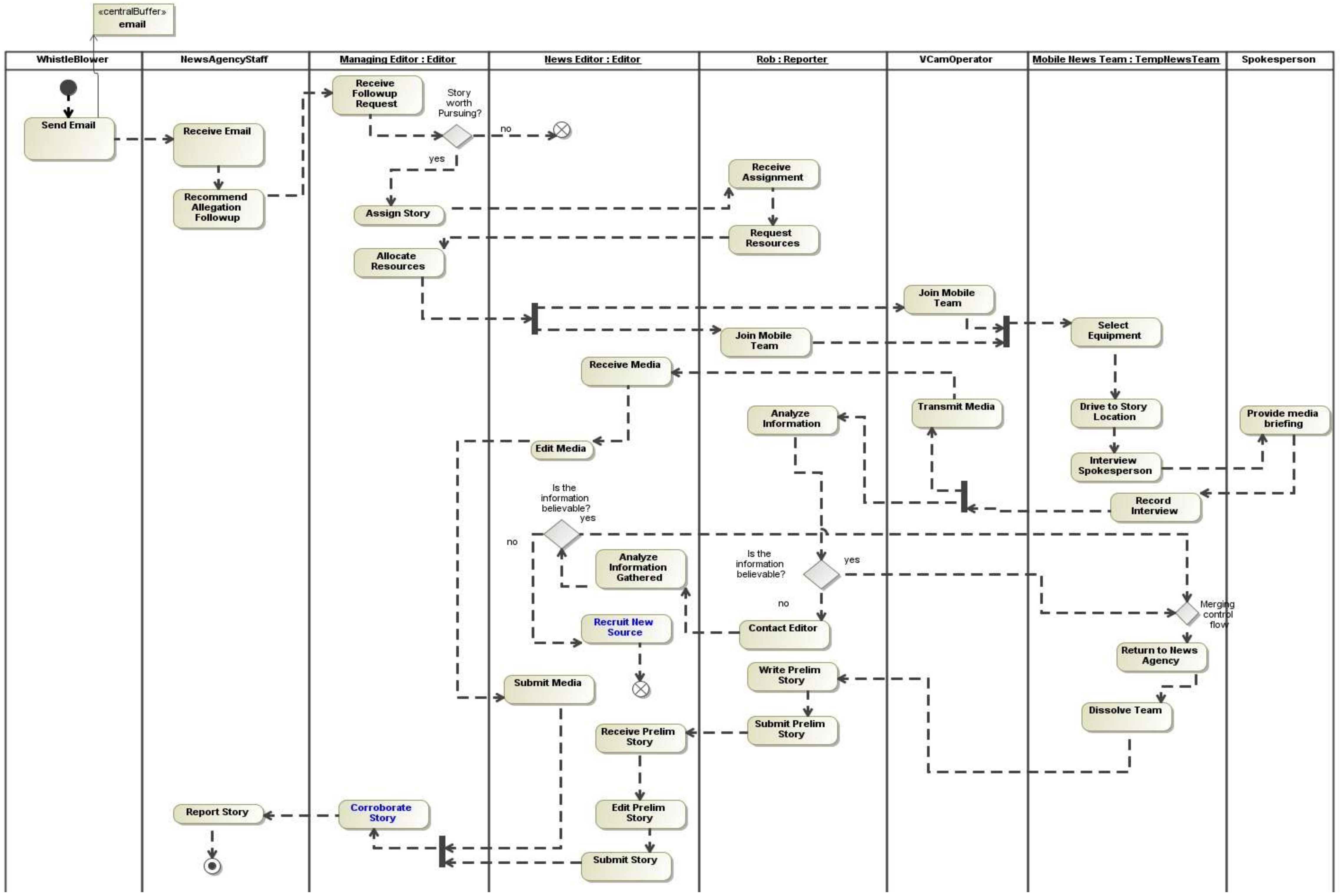


Figure 51 News Agency Scenario 1

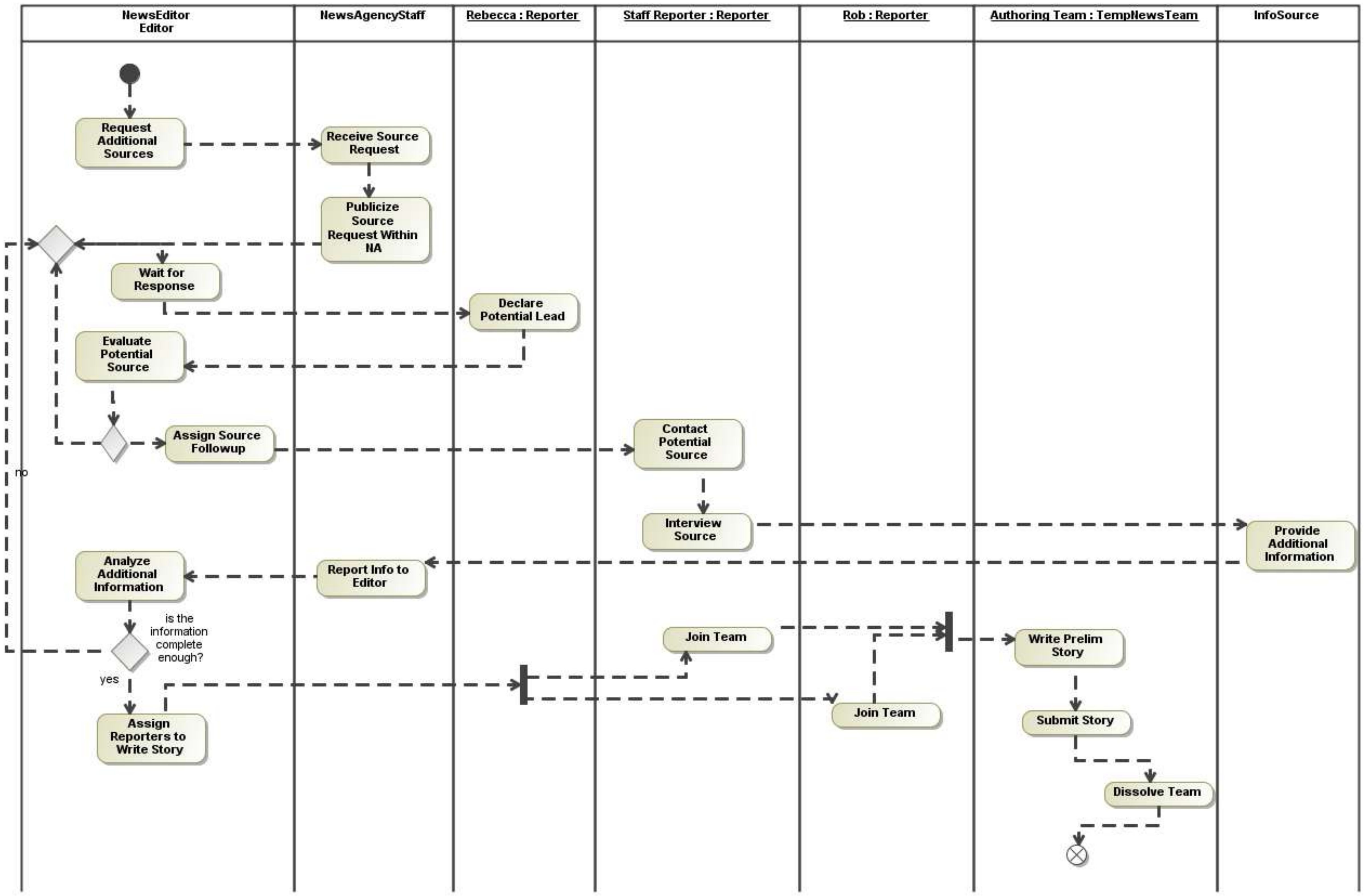


Figure 52 News Agency Scenario 2

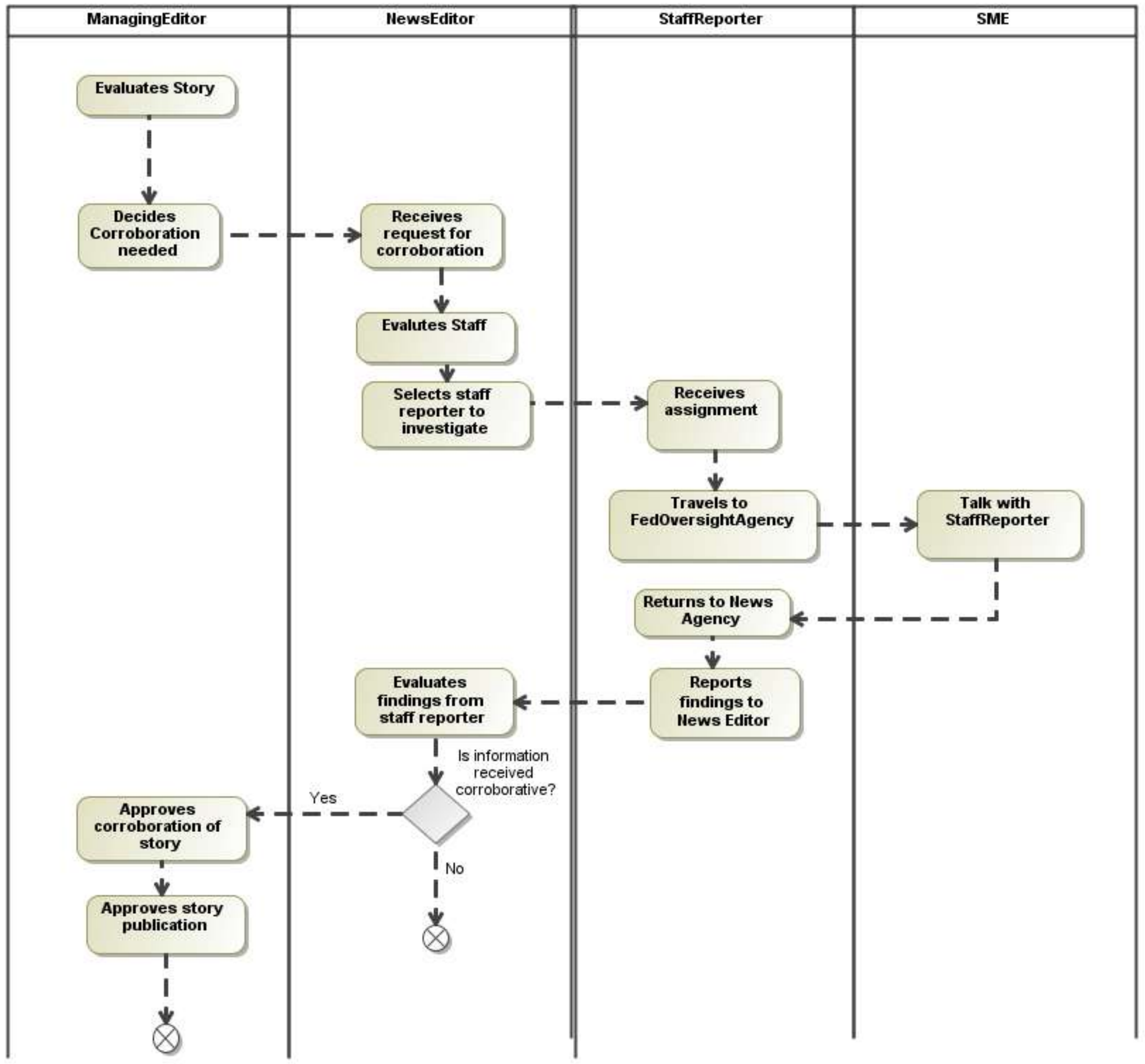

ManagingEditor
NewsEditor
StaffReporter
SME
Evaluates Story
Decides Corroboration needed
Receives request for corroboration
Evalutes Staff
Selects staff reporter to investigate
Receives assignment
Travels to FedOversightAgency
Talk with StaffReporter
Returns to News Agency
Reports findings to News Editor
Evaluates findings from staff reporter
Is information received corroborative?
Yes
No
Approves corroboration of story
Approves story publication

Figure 53 News Agency Scenario 3

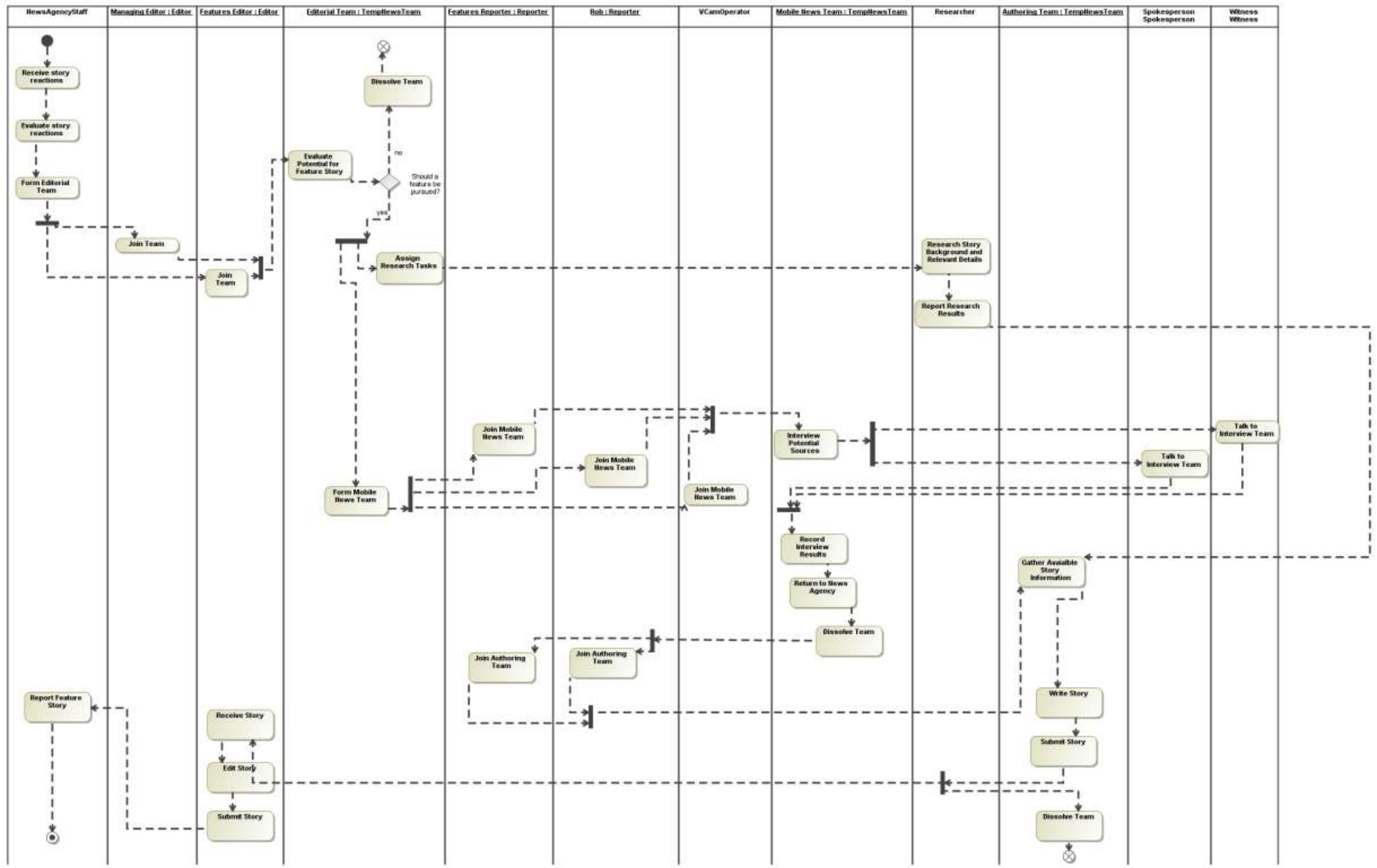


Figure 54 News Agency Scenario 4

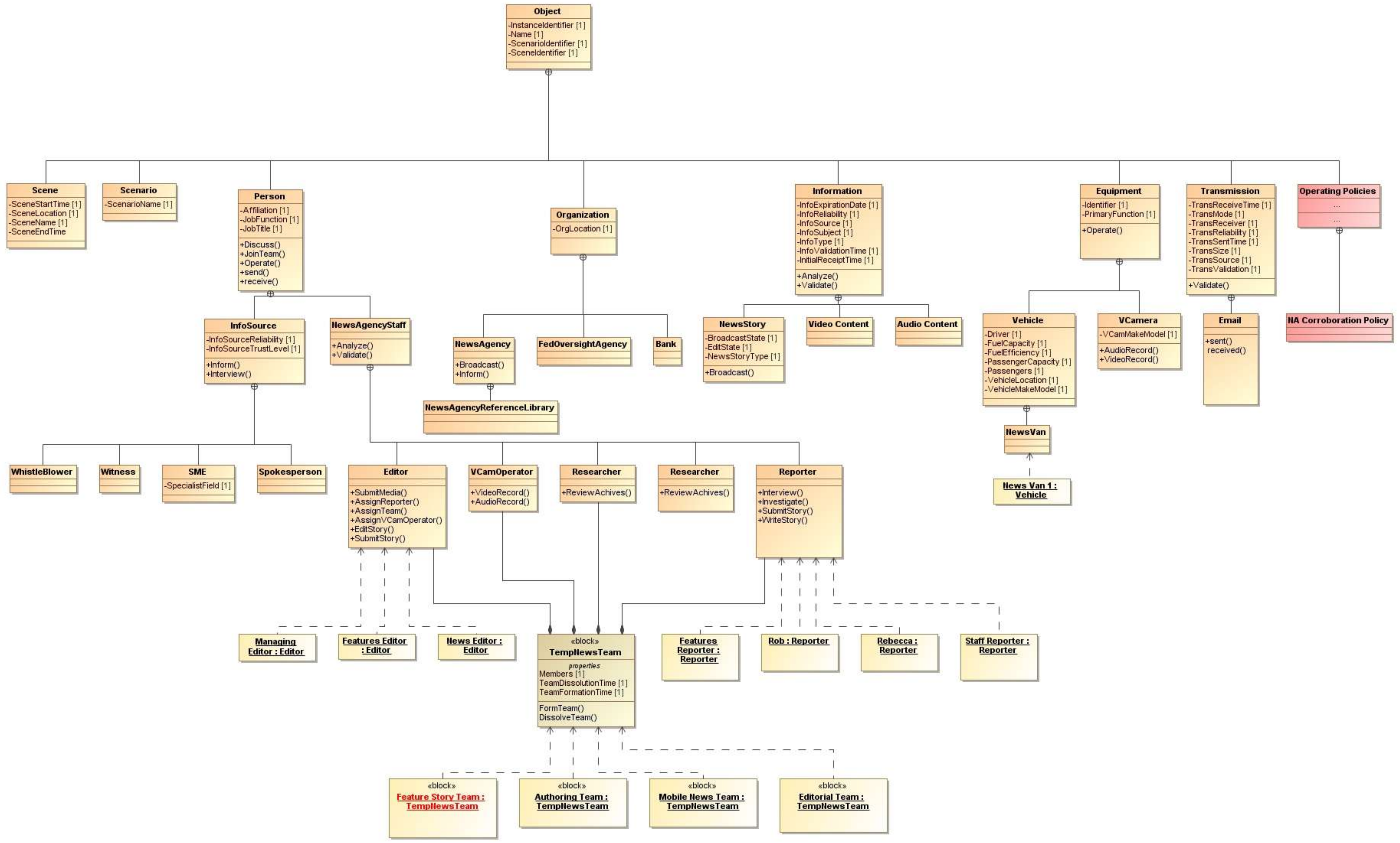


Figure 55 News Agency Scenario Primitives

7.2 Workshop Results

During the December workshop, participants were divided into four team four teams, with an average team size of four people each and each were provided a laptop and the ICEF prototype. The participants were provided Scenarios 1 and 2 as presented above. The collection of primitives programmed into the prototype was able to allow a full mapping of Scenario 1, while successfully representing Scenario 2 in ICEF would require some level of adaptation of the existing primitives and ICEF capabilities.

The groups were imaginative in their use of the software. Since it was a prototype, some of the functionality was limited and this led the users to push the software envelope quite a bit. Some of the observed activities included:

- re-purposing less mature existing 3-D objects and using them as placeholders in scenarios
- blurring the lines between what the tool allows users to do vs. what users want to do
- developing new scenarios, unrelated to the ones originally distributed for testing

7.2.1 Evaluator Feedback

The feedback received from the workshop participants fell into three categories; positive aspects, detractors and suggestions for future development. Table 8 recounts some of the feedback collected:

Table 6: ICEF Evaluator Feedback

Positive
• the graphical representation and immersive environment led to a more realistic approach to CONOPS development • the idea of links and groupings was applicable for use in the evaluators' working environments • the immersive environment makes it easier to see anomalies and contradictions, and also makes it more enjoyable to visualize the scenario
Detractors
• difficulty was encountered specifying links between objects and "intangible objects" such as organizations • the software at its present level of immaturity did not lend itself to extension into other scenarios • there was no persistence of objects, since database was not implemented at this version

Suggestions/Observations

- add a notes section, to allow user annotation
- provide multiple camera views/perspectives
- graphically display links, rather than simply listing them
- filter links and objects, in order to sort and display only those elements of interest
- drag and drop objects from list
- provide "stubs" for objects, attributes and links, to enable authors to insert placeholders for missing or incomplete information
- provide animation for scene transition, if applicable
- physical space is limited, perhaps allow modelers to define their own terrains, maybe using real-world mapping interface
- develop new ontologies and taxonomies for different domains
- non-graphic concepts will need to be consistently represented in the environment

Feedback collected from workshop participants and sponsors was used to drive the goals and objectives of the next phase of this research, which will be discussed at the end of this report. Detailed data analysis is found in Appendix B.

7.3 RT 30 Research Survey and Analysis

To begin to study the effectiveness of the ICEF, an extensive literature review was conducted to discover metrics that exist for evaluating concept engineering tools and processes. While a fully formed metrics and evaluation scheme that fit the needs of this research had not been previously created, there was considerable investigation of assessment techniques for collaborative problem solving, as well as indicators of shortcomings of CONOPS that need to be addressed. Given this research, a set of metrics was developed to assess concept engineering. These metrics were separated into:

- Artifact metrics – to enable CONOS-specific assessment

- Collaboration metrics – to study how users and engineers work together to develop a CONOPS
- Experience metrics – to measure the effect that different professional and life experiences have on CONOPS development

The metrics are summarized below (Table 7, Table 8, and Table 9) and are further described in (Korfiatis, 2013). Each metric was used to derive a survey question to be delivered during CONOPS experiments, discussed below.

Table 7: Artifact metrics

Metric/Source	**Definition**	**Survey**
Understandable	How easy it is to understand artifact	• The final artifacts provide a sense of the overall scenario • The final artifacts are easy to understand
Balanced Point of View (IEEE, 1998)	How well the artifacts represent a collection of individual PoVs How well do users express expectations through the artifact	• The final artifacts represent an acceptable balance between all of the group's needs • The final artifacts favor the needs of one stakeholder over another
Accuracy	How accurately artifacts represent scenarios. CONOPS must provide accurate descriptions of needs	• The final artifacts clearly represent needs of your role • The final artifacts clearly represent an accurate portrayal of the group's negotiated scenarios
Applicability to System Design	How useful the artifact would be to future developers A textual document tends to be cumbersome and of little use as a communication tool between stakeholders and developers	• The final artifacts provide clear guidance to system designers for system development • The final artifacts provide a useful tool to promote future conversation between stakeholders • The final artifacts be useful for educating new stakeholders later in the development process
CONOPS Elements (Fletcher, 2012; Roberts, 2008; Saldana, 2012)	Does the artifact include CONOPS elements that are required in CONOPS standards but shown to be under-addressed in traditional CONOPS?	• The final artifacts represent human roles • The final artifacts clearly represent the number and type of personnel required for scenarios • The final artifacts clearly represent personnel interfaces in the scenarios
Maintainability and Evolve-ability	How easily the artifact could be maintained, updated or evolved CONOPS should be updated to reflect evolving situation Amending textual CONOPS can be time consuming and lead to inconsistency across document	• The final artifacts are easy to edit if stakeholder needs were to change • The final artifacts are easy to edit to address new stakeholder needs

Table 8: Collaboration metrics

Metric/Source	**Definition**	**Survey**
Reduce Development Time (Mostashari, McComb, Kennedy, Cloutier, & Korfiatis, 2011)	Time required to develop CONOPS	• The time required to produce the scenarios is reasonable given the quality of the final artifacts
Satisfaction with Collaboration	Indicates that stakeholders leave with a sense of satisfaction that the collaboration was effective	• You are satisfied that the final scenarios address your need. • You are satisfied with the collaborative exchange during scenario development
Mutual Understanding (D. F. Noble, 2004)	The extent to which team members agree or disagree	• Your needs were adequately understood by the group • You adequately understood the needs of other group members • During scenario development your groups was able to correct misconceptions on each other's need
Communication (Fletcher, 2012) (Linebarger, Scholand, Ehlen, & Procopio, 2005)	How was communication between team members affected by the use of a specific scenario development process	• The scenario development process led to clear and unambiguous conversation about the scenarios • The scenario development process promoted critical dialog and skepticism • Verbal communication was improved through using the scenario artifacts
Shared Mental Model (Cloutier et al., 2010; McComb, 2007)	A common internal representation of the world, an event or a user scenario that is shared between team members	• A shared vision of the problem was reached by the group during the development process • A shared vision of the solution was reached by the group during the scenario development process • A shared vision of typical user scenarios was reached by the group during the scenario development process

Group Problem Solving (D. Noble & Kirzl, 2003)	Group problem solving is a major subset of team activities presented in the Framework for Collaboration Model (D. Noble & Kirzl, 2003). Five major types of group interaction can take place during group problem solving	**Brainstorming** • Adequate consideration was given to alternatives during scenario development • By developing the scenario artifacts, alternatives were discussed that would not have been otherwise discovered through conversation • A broad range of solutions were considered by the group that were relevant to scenario development **Prioritization** • The scenario development approach helped the group explicitly prioritize stakeholder needs • Any prioritization that took place during scenario development is reflected in the final scenario artifacts • The scenario development approach helped the group implicitly prioritize stakeholder needs **Discovering differences** • During scenario development, fundamental differences in stakeholder needs were discovered and discussed • By developing the scenario artifacts, differences that were discovered that would not have been otherwise discovered through simple conversation **Negotiation** • By developing the scenario artifacts, there was more opportunity for meaningful negotiation than there would have been through simple conversation • Each stakeholder was able to fully present and explain their needs during scenario development. • One stakeholder's needs dominated the scenario development process **Consensus** • The scenario development process has allowed our group to reach a consensus on scenarios that everyone agrees with. • By developing the scenario artifacts, there was a greater level of consensus in the final scenarios than there would have been through simple conversation
Collaborative Macro-Cognitive Process (Letsky, Warner, Fiore, Rosen, & Salas, 2007; Warner, Letsky, & Cowen, 2005)	"the internalized and externalized high-level mental processes employed by teams to create new knowledge during collaborative problem solving" (Letsky et al., 2007)	**Adapted from Warner et al. (2005) to be captured and codified by observers:** • Visualization & Representation - presenting information in pre-processed forms • Building Common Ground - sharing common or joint knowledge and beliefs to build common ground (• Knowledge Sharing and Transfer - information is given by one person and received by another • Team Shared Understanding - synthesis of essential knowledge, held collectively by some and/or all team members • Solution option Generation - generating set of decision alternatives that satisfy the requirements of the task • Negotiation of Solution Alternatives - discussion to construct something new which neither individual could create on their own • Team Pattern Recognition - process of recognizing patterns in information, solution options or problem space • Converge individual mental model to team mm - convincing others to accept your data, information or knowledge • Critical Thinking - reflective reasoning about beliefs and actions • Mental Simulation - using mental models to make inferences about future states of a situation (what if…) • Intuitive decision making - A team rapidly reaching intuitive consensus • Compare solution against goals - discuss a final solution option against the goal • Analyze and Revise solution Options - analyze final solution options and revise them if necessary

Table 9: Experience metrics

Metric	Definition	Survey
Gaming Experience	Level of comfort in playing or creating video games or gaming engines, 3D immersive environments or other advanced visualizations	Rate your comfort with the following concepts/activities: • Game playing • Game development • Visualization
Systems Engineering Experience	Work experience and level of comfort in systems engineering related activities	• How many years of experience do you have in systems engineering? • How many systems engineering projects would you estimate you have been involved in? Rate your comfort with the following concepts/activities: • Systems Engineering • Model Based Systems Engineering • System Design • Modeling and Simulation
CONOPS Experience	Exposure, work experience and level of comfort to CONOP documents and CONOPS/Concept Development	• How many CONOPS development processes have you participated in? • How many CONOPS documents have you read? Rate your comfort with the following concepts/activities: • Concept Development • CONOPS Development • Requirements Elicitation/Management

7.3.1 ICEF Experimental Procedure

Two laboratory experiments were conducted to study the effectiveness of ICEF. Both experiments involved participants producing artifacts representing the operational scenario section of the CONOPS document. All groups were presented with a number of written descriptions of a news agency scenario and asked to either model operational scenarios using the ICEF tool or create a text based narrative akin to the traditional CONOPS development process. Due to limitations placed on the experimental design by theRT30 research sponsor, there was no control group for the first experiment. Attendance in this first experiment consisted of twenty-one DoD systems and software engineers, development and operations personnel, technical writers, and managers separated into five groups. The experiment was conducted as displayed in the SysML activity diagram in Figure 56.

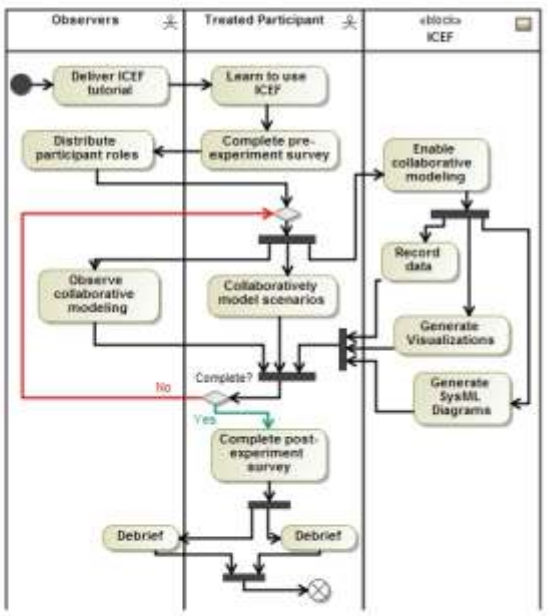

Observers
Treated Participant
«block» ICEF
Deliver ICEF tutorial
Learn to use ICEF
Distribute participant roles
Complete pre-experiment survey
Enable collaborative modeling
Record data
Observe collaborative modeling
Collaboratively model scenarios
Generate Visualizations
Generate SysML Diagrams
Complete?
No
Yes
Complete post-experiment survey
Debrief
Debrief

Figure 56 ICEF experiment 1 procedure

A brief instructional tutorial on the functionality of the ICEF tool was presented to all participants by the project manager and a primary ICEF software developer. After fifteen minutes of instruction, all participants felt comfortable with the functionality of ICEF and a pre-experiment survey was administered to record the participants' level of experience in CONOPS development, gaming, visualization and systems engineering. Following the completion of the pre-experiment survey, each participant received a participant instruction handout, which provided general instructions on the experiment, background information on typical operation of a news agency, and a description of the five specific scenarios to be modeled. A list of roles (news editor, reporter, systems engineer, support asset manager, and acquisitions and support personnel) was provided in the participant instruction handout. The groups were allowed to assign roles using any method in which they decided and each participant was provided with a role specific handout. These handouts were written to inform each participant with the needs, concerns and responsibilities of their roles, which were purposely set to be contradictory.

Once the role-specific handouts were distributed, the experiment began. Each group was responsible for collaboratively modeling as many of the scenarios as they could manage using ICEF. Their primary objective was to be sure that their roles' needs and concerns were evident within the model. Because the first experiment took place using DoD employees, audio recording of the modeling sessions was not possible. Therefore, during the first experiment, five observers used a structured observer rubric to evaluate the type of collaborative cognitive processes each group was undergoing. At the end of the session, each team produced animated visualizations and SysML diagrams featuring the operational scenarios as modeled using the ICEF. Finally, participants were asked to complete a post-experiment.

The second experiment was run in a similar fashion using twenty-five Engineering Management undergraduate students from a third-year Engineering Design class. While not active in the systems engineering domain professionally, these students had recently concluded coursework related to CONOPS development and requirements elicitation taught by a systems engineering professor with numerous years of practical industry experience. The students were separated into eight groups. After being divided, four of the groups were asked to move to a separate room where they were subjected to the same experimental procedure utilized in the first experiment. The remaining four groups acted as control groups and did not receive information about the ICEF. Instead, they were asked to develop a textual description of the same five news agency scenarios. The result of their discussions was a Microsoft Word document containing a narrative describing the operational scenarios, comparable to the current CONOPS artifact and development process. The use of student allowed for audio recording of the sessions, and as such, the research team recorded conversations by each of the groups. After the experiment, the same five observers were asked to listen to the recordings and codify the types of macro-cognitive collaborative processes taking place during the groups CONOPS development session. A detailed look at the second experimental procedure is seen in Figure 57.

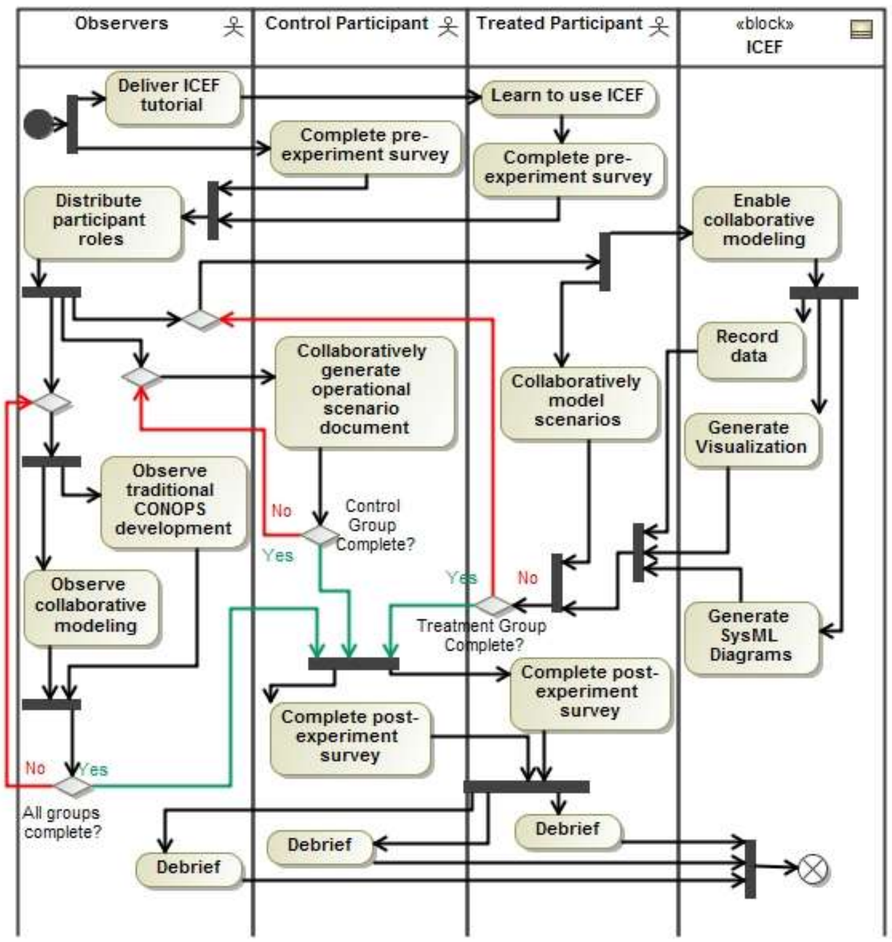

Observers
Control Participant
Treated Participant
«block» ICEF
Deliver ICEF tutorial
Learn to use ICEF
Complete pre-experiment survey
Complete pre-experiment survey
Distribute participant roles
Enable collaborative modeling
Record data
Collaboratively generate operational scenario document
Collaboratively model scenarios
Generate Visualization
Observe traditional CONOPS development
No
Yes
Control Group Complete?
Observe collaborative modeling
Yes
No
Treatment Group Complete?
Generate SysML Diagrams
Complete post-experiment survey
Complete post-experiment survey
No
Yes
All groups complete?
Debrief
Debrief
Debrief

Figure 57 Experiment 2 activity diagram

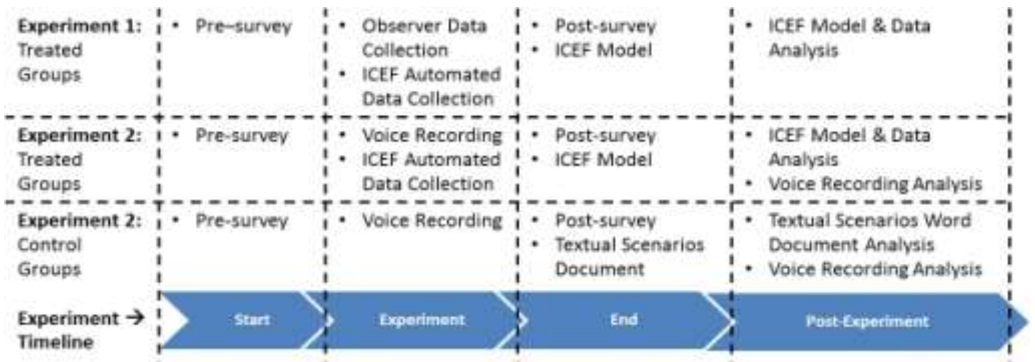
7.3.2 Data Collection

As briefly described above, data was collected during the experiment using three methods (Figure 58).

- Surveys were used to elicit feedback directly from experiment participants. To establish a baseline on the background, expertise and comfort level of participants in the fields of systems engineering, CONOPS development and gaming, visualization and immersive environments, the pre-experiment survey asked participants to select a pre-determined range of values describing their years of systems engineering experience, the number of systems engineering projects they have worked on and the number of CONOPS they have read and developed. Participants also ranked their experience in topics related to this research as Very Experienced, Some Experience or No Experience. A post-experiment survey was also distributed to evaluate ICEF's effectiveness. The survey was designed to assess the participant's perception of the collaborative modeling process and the resultant CONOPS artifacts.

- The method of observation used in this experiment was grounded in established models for measuring cognitive processes during collaboration. Based on previous collaboration research, observation was centered on classifying the types of macro-cognitive processes used by participants during the collaborative scenario modeling. The collaboration model used attempts to measure how many instances of specific cognitive processes occur, how often they are encountered and when they transpire. To reduce possible observer bias, each group of participants was observed by two observers at a time and the observers rotated groups every twenty minutes. Differences in scoring were discussed and reconciled between the observers. Additionally, the database and logging function of each ICEF system provided researchers with the ability to recreate and document what occurred in the software while specific macro-cognitive processes were taking place. Observers were also responsible for collecting qualitative observations of individual and group behaviors during collaboration.

- The ICEF was specifically designed to capture information regarding how the users interacted with the software. This was carried out using internal activity logging. The activity log serves a number of purposes including measuring timing between modeling activities and recording the placement and deletion of objects, relationships and attributes.



Experiment 1: Treated Groups	• Pre–survey	• Observer Data Collection • ICEF Automated Data Collection	• Post-survey • ICEF Model	• ICEF Model & Data Analysis
Experiment 2: Treated Groups	• Pre-survey	• Voice Recording • ICEF Automated Data Collection	• Post-survey • ICEF Model	• ICEF Model & Data Analysis • Voice Recording Analysis
Experiment 2: Control Groups	• Pre-survey	• Voice Recording	• Post-survey • Textual Scenarios Document	• Textual Scenarios Word Document Analysis • Voice Recording Analysis
Experiment → Timeline	Start	Experiment	End	Post-Experiment

Figure 58 Experiment data collection

7.3.4 Research Experiment Hypotheses

CONOPS can be examined in terms of both collaboration during development, and the final artifact. To this end, two hypotheses were developed to drive data collection and analysis.

- Hypothesis 1: Use of the ICEF will improve the operational scenarios artifact of a CONOPS.
- Hypothesis 2: Use of the ICEF will improve collaboration during the development of the operational scenarios section of a CONOPS.

7.3.5 Bayesian Data Analysis

Given the scope of this research and the design of experiments described above, there were limiting factors in selecting an appropriate data analysis technique. These include the fact that:

- few recognized metrics have been established or data collected and published concerning CONOPS development and concept engineering
- data collection lead to both qualitative and quantitative data so the analysis technique should be able to handle both types of data
- the sample size of the experiment was relatively small and was not fixed across experiments

Given these limitations, Bayesian Hypothesis Testing was selected for data analysis. In-depth discussion of Bayes' theorem and Bayesian data analysis is beyond the scope of this report. A full treatment of Bayesian data analysis can be found in (Fenton & Neil, 2012; J. Kruschke, 2010).

Bayesian analysis was selected here because it is fairly accurate with smaller sample sizes (Uusitalo, 2007), it does not require a fixed sample size across experiments (J.K. Kruschke, 2010) and it is effective in combining both experimental and observation data (Cooper & Yoo, 1999). Additionally, since there is little previous published work on concept engineering, the Bayesian approach is well positioned to capture this uncertainty and treat it explicitly (Uusitalo, 2007).

As described in (Fenton & Neil, 2012; J. Kruschke, 2010), Bayesian hypothesis testing is easily conducted using a Bayesian network. A Bayesian network is a directed acyclic graphical representation of a set of variables and their relationship to each other. The network nodes can represent variables, parameters, hypotheses or observed data, and the directed edges describe the relationships between these variables. In Figure 59, the metrics described above were added as middle level nodes in the Bayesian network (green nodes). Specific data from surveys and other sources were added as input nodes (yellow nodes). Two nodes were created to represent each hypothesis, and connected to a final output node labeled Combined Output (orange rectangular nodes).

For this research, due to lack of established prior data, it was assumed that the experiments described above would result in one of three distinct outcomes, each of which can be seen as a competing hypothesis:

- Alternative Hypothesis 1 (H_{A1}) - The observed data shows evidence that the ICEF was ineffective in improving CONOPS artifacts and collaboration
- Alternative Hypothesis (H_{A2}) - The observed data cannot be used to make a judgment as to the effectiveness of the ICEF in improving CONOPS artifacts and collaboration
- Alternative Hypothesis 3 (H_{A3}) - The observed data shows evidence that ICEF was effective in improving CONOPS artifacts and collaboration

Based on (J. Kruschke, 2010), if the data gathered from a group using the ICEF is inputted to the Bayesian network and the resulting probability distribution:

- fits entirely within the H_{A1} distribution, the data has a high probability of supporting the conclusion that ICEF was ineffective
- fits entirely within the H_{A3} distribution, the data has a high probability of supporting the conclusion that ICEF was effective

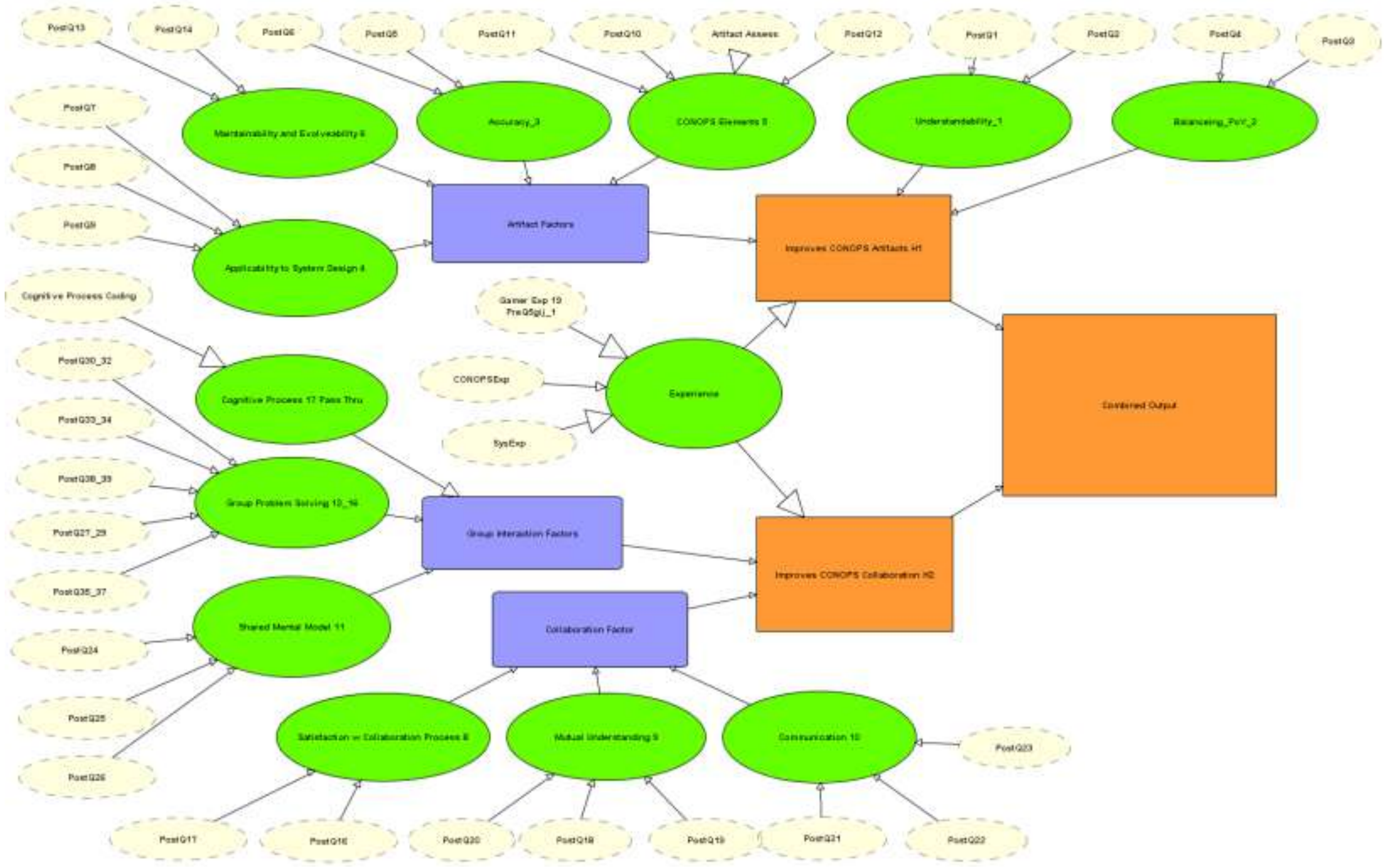


Figure 59 ICEF Bayesian network

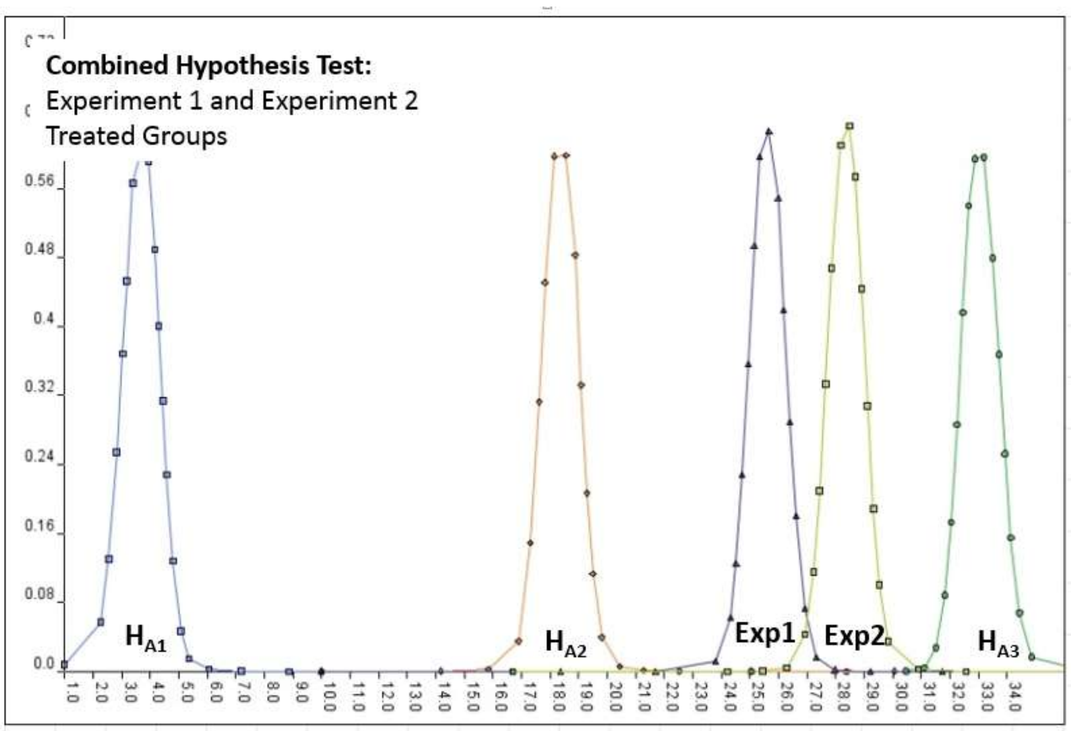


Figure 60 Bayesian hypothesis test

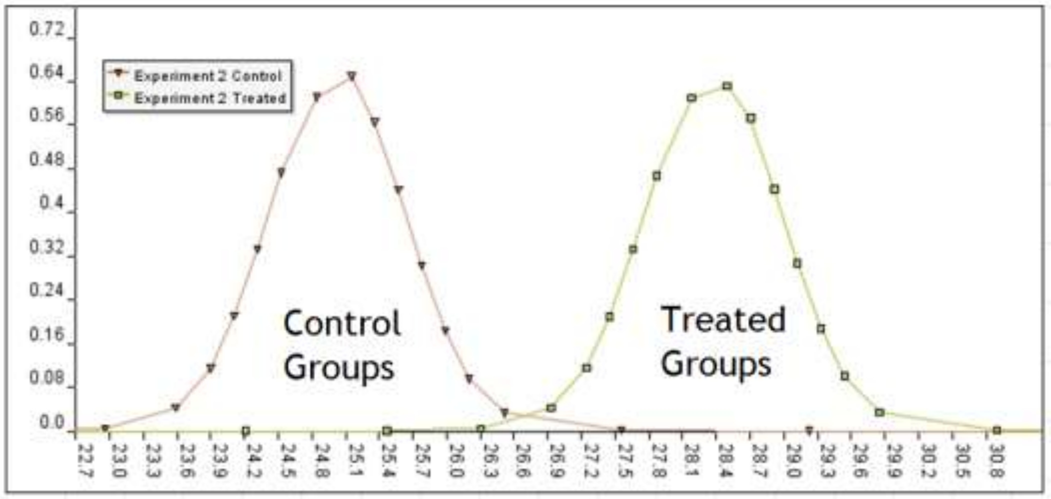


Figure 61 Experiment 2 comparative analysis

7.3.6 Experiment Results

Figure 60 shows the results of the Bayesian analysis using data collected from both treated groups in the first and second experiment. Each set of observed data formed a probability distribution that lies between H_{A2} and H_{A3}. Because the distribution does not fit well within any hypothesis distribution, none of the alternate hypotheses developed can be accepted with full confidence. However, Bayesian hypothesis testing demonstrates coherence (Wagenmakers et al., 2010), meaning that the position of each distribution can be directly compared to each other, and conclusions can be drawn based on the relative position, size or shape of a distribution. From Figure 60 we can see that the distributions of the observed data fall very far outside the distribution of H_{A1}. It can be stated with confidence that based on the data collected from these experiment there is little to no evidence in favor of accepting H_{A1}, and it is far more likely that evidence exists to support H_{A3}. While this is not an outright acceptance of any alternative hypothesis put forth, the data collected in these experiments are much more likely to support the effectiveness of ICEF rather than its ineffectiveness. The proximity of the observed data's distribution to the H_{A3} posterior is an indicator of the level of confidence of this conclusion. At the same time, Figure 61Figure 61 displays a comparison between the treated and control groups of the second experiment. The distribution of those who received the treatment (utilized the ICEF) and those who acted as the control group (did not utilize the ICEF) can be compared to one another directly. Based on this comparative analysis, the data from this experiment shows a preference for the ICEF approach over the traditional concept engineering approach.

8.0 Lessons Learned

There were many lessons learned during the original phase of research (RT30). They remained consistent across the most recent phase of the research, and are presented again, along with several newer topics engendered by the interim workshop. They are grouped by topics: Research, Project Management, System/Software Architecture, and Project/Code Construction. As important as the listing of lessons learned during this research is, our solutions to these situations are of interest, too, and are outlined in italics beneath each lesson statement (whenever applicable).

8.1 Research Lessons/Questions

1. The software effort is actually secondary to the research questions being asked, but consumes 98% of effort in the early stages – keep the research questions at the forefront of each member's attention.
   - *We selected a combination of active and passive reinforcement of this thrust. The phrase "What are the research questions?" was posted on every white board, and was continually invoked during meetings whenever new features, alternate strategies, or additional interfaces were discussed.*

2. Can the process of CONOPS development and understanding be improved through the use of a graphical user interface?

3. What's missing in a graphical scenario building paradigm?

4. What's gained from the graphical scenario building paradigm?

5. Does real-time collaboration between distributed stakeholders improve the CONOPS development?

6. Can a real-time collaboration environment enable quicker consensus on CONOPS generation?

7. Are there new or specific issues in asynchronous software development in an immersive environment?

8.2 Project Management

1. It is critical to continuously monitor migration to new development environment releases – we now only migrate as a team, and then only after testing current builds in new release.
   - *This strategy has worked well in the current development environment. Both the Stevens and Auburn teams coordinated this effort.*

2. Iterative development tasks must be clearly defined/described to avoid redundant efforts and conflicts; this takes a lot of time and effort

3. Use of Agile processes is very difficult in academia – neither students nor faculty are regular in their schedules/work times.
   - *This particular problem was exacerbated this phase of research by the increased number of students involved. Finding available times for group meetings became much more difficult with varying class schedules for the students, and teaching schedules for the researchers. Additionally, disparate university schedules also impacted participation. There may be no easy solution to this.*

4. Because of the above, the use of Skype and Google+ hangouts were tested, hoping they'd be effective, especially when it came to review, walk-through, testing, and team-effectiveness. Going forward, we would initiate daily meetings of short duration, to assess progress and discuss modifications.
   - *As stated above, although we'd like to initiate the daily stand-up meeting paradigm, it was virtually impossible for the diverse student staff of the participant universities. As discussed above, there may be no easy solution, except for a mandated daily, meeting time for key members.*

5. International of workforce composition (students from various backgrounds and cultures, with English not necessarily their preferred language) has its own challenges.
   - Clear Skype or Google+ hangouts, or even written word communication can be challenging - clarity can suffer when there's a lack of visual clues
   - Video conferencing is highly preferred over voice-only or written communication
   - Face-to-face remains the best way to manage, but video is a valuable 2nd best
   - Avoid idioms when describing scenarios and scenes
   - Analogies can work, but should be simple and clear

6. Measuring progress via visible functionality is not helpful, nor is using long-standing measures such as SLOC
   - Other criteria must be adopted and we propose a combination of SLOC count and a partitioning of categories of code: infrastructure, actual 3D object presentation, and 3D model manipulation
   - Current Statistics
     - SLOC for final executable: XXXX, # objects: 199

7. Organization of code within the project listing is critical, especially when using a multiple-developer approach
   - Naming folders clearly is helpful
   - Grouping scripts together is critical, since most of the scripts are small (and again, naming clearly here is also critical to efficient searches)
   - Clarity among the team members is paramount, along with clear naming rules and conventions
   - *The institution of basic standardization did help, in an overall sense, but it emerged that the source of difficulties arose from the amount of professional experience coders had. This differed among both team members within a site, as well as between the schools. Documentation helped reduce about half of these issues, but there is no substitute for industrial coding experience.*

8. The actual 3-D models used are free for academic and research use but are not free for commercial distribution; this is a consideration if deploying in an organization, so factor time for 3-D model development

9. The use of Trello, an online project management tool, was invaluable in managing task assignments between the two programming locations. By attaching tasks to various build cycles, each team was able to stay on target with their deliverables.

10. Weekly meetings were helpful not only in tasking but when discussing difficulties encountered by either team. It is strongly suggested that multi-site development always adopt a weekly review of both code and project plan.

8.3 System/Software Architecture

1. Evaluation of scenarios was a key factor in an entire redraft and reconstruction of the architecture – representation of time (and activity ordering) became critical and factored into our final interface look and feel

2. All architecture and data objects should be specified as completely as possible and as early as possible

3. Because architecture (and infrastructure) is not "seen," the work done is not obvious to management/ customer – and therefore it gives the impression of "no progress"

4. Error handling architecture should be context-reliant, and needs to be addressed for consistency

5. A subtle concept came to light – that is, the common formation of temporary teams for moving activities forward in a scenario. Having these "primitive groups" become objects with a life of their own requires new storage and naming facilities because the authors now, in effect, become developers – blurring the lines between the responsibilities and authorizations

6. In addition to the tracking of temporary groups in terms of data, considerations of representation also arose - these issues have been deferred to the next phrase; during the development of animation (or execution) of the scenes, temporary objects undergoing transformations (such as containment) will need a consistent strategy

7. We need to develop an ontological schema similar to semantic webs (such as OWL), and will research this moving forward

8. Representation of links is currently via a list, although it should also have a visible component, this was an issue because of potential graphical crowding - the use of filters was proposed and will be investigated moving forward

9. One major addition during this phase of the project was the integration of the RT-31a scenario into the existing architectural framework used by RT-30a. Several items arose during this integration:
    a. Performance was not a strong focus here, proof-of-concept was the driving consideration.
    b. In the process of integration, it was inevitable that some refactoring of code would occur. This required caused additional test time to validate the robustness of the code.

c. The domain of RT-30a, which of necessity had several differing locales required the use of multiple cameras. RT-31a, in contrast, is a single locale viewed across several time periods. This alternate view of a scenario required additional modifications to the data structure as well as the overall architecture.

8.4 Project Code/Construction Lessons Learned

1. Individual project access in asset server needs to be transparent to all developers

2. Managing Asset Server took more time than anticipated, and there was no easy way to roll-back to a previous release or version

3. There were occasional slow-downs when committing to or updating from the Asset Server

4. Highly-modular design vies with programming strategies – optimal breakpoints must be developed

5. Assignment of modular design elements is also problematic and, because of the iterative nature of design and development, is a real challenge

6. Graphic design of 3D models and manipulation of them, and scaling took longer than originally anticipated; this is not due to the provenance of the models, but is inherent to 3D environments

7. As in all the Unity-based software, programmers are encouraged to avoid manipulation of the object surface meshes – in order to indicate a "selection" insert an indicator above the object itself

8. Avoid manipulation of the object surface meshes – in order to indicate a "selection" insert an indicator above the object itself

9. Along with time being a critical design component, the idea of simultaneous activity representation is also implicit in scenario building; we will investigate the accommodation of simultaneous activities in the next phase of design

10. An object drop default strategy is needed even if one is able to drag/drop objects – user-selected positioning is ideal

11. Movement of groups (for instance, a group of passengers entering a vehicle or building) will be crucial as the design moves forward, in terms of visual representation, as well as generated output

12. The movement of groups mentioned highlights another issue, that of containment of objects.

13. Manipulating colliders is how objects have solidity in the environment - this will be an important consideration when making scenes executable

14. Consider the use of multiple cameras as a default mechanism for each scene when being built, and provide an easy toggle for users to switch views

15. Asset Server Change Control/Staging Platform
    - Individual project access in asset server still needs to be transparent to all developers.
    - There were frequent slow-downs when committing to or updating from the Asset Server, this was especially noticed by the Auburn team.

16. Actual physical movement of groups (for instance, a group of ground vehicles, personnel, or deployment of fleet) is quite intricate. Additional time is needed to coordinate all the movements and storage of that data.

17. Containment of objects can impact performance, storage, and retrieval. Due to the small number of objects manipulated, this wasn't seen in these applications, but it may become a larger issue for more populated scenarios.

18. Drag and drop functionality was implemented to allow for easy placement and movement of objects in the scene. The user was given the ability to move the camera around at their discretion using the "W, S, A, and D" keys. The viewing angle could be more easily controlled by the mouse. This gave the user a great deal more ability to navigate the CONOPS environment.

19. Project coding standards, naming, and organization became more critical, especially since the development was shared between two geographically-diverse coding teams, with differing levels of Unity experience. This was seen in both

parallel development and with the code management provided by the Asset Server.

20. Although RT30a and RT31a now share a common framework and architecture, the domain and thrust of analysis was different for each research task. Although the sample space of experience is small (2 domains), it is clear that adding a second, somewhat similar domain type does not noticeably lengthen development time. If the environment being modeled is radically different physically (underwater, atmospheric) this may not be the case, but for these environments, it was not excessively time-consuming.

21. For RT31a, the time-consuming work resulted from the somewhat specific considerations of the domain – being one of a combat contact-to-fire scenario rather than a collection of street scenes. Verb lists, interactions, the fluid flow of scenes, and the specifics of military collaboration/conflict (time-dependent and activity-dependent health and supply monitoring) required additional design and display considerations. To have a common code base, this had to be in the data model, though many domains may not use it… this will probably be true for each new domain added – the domain ontology will grow with each new domain for domain specific terms and capabilities. Further research would be necessary at this point whether it is better to have a single database for all the domains, or construct smaller databases for each new domain added.

22. When looking at domain-initiation considerations, it was necessary to examine the loading of 3D models dynamically during run-time. It was determined that Unity does support the importation of objects at run-time via the use of Asset Bundles. These are a Unity file type that consists of grouping of like files belonging to a single object (such as a 3D model of a soldier complete with animation, wireframe, color palette, meshes, etc.).

23. There is a distinct lack of free high-quality 3D models available, that are also capable of sophisticated motion. This is not true of static "window-dressing" models, or wall treatments. Care must be taken if the sponsor plans to distribute models, whether free or purchased. This may also impact further development if this effort is transitioned to an open-source environment.

9.0 CONCLUSIONS

There are a wide range of conclusions that can be drawn from the thousands of man-hours invested in this research and development endeavor. Many are related to software engineering, distributed academic collaboration and game development for systems engineering. However, the primary tasking of this research project has been the investigation of methods, processes and tools for the advancement of concept engineering. As such, conclusions made here will focus on the research at hand, and peripheral conclusions will be reserved for future publications.

The goal of this research task has been to develop a concept engineering software demonstrator that enabled the analyst, engineer, or warfighter (depending on which RT one looks at) to generate operational scenarios through a process, and in a medium, that resulted in an improved shared mental model during early systems engineering activities. The work of this phase of research was to extend the Integrated Concept Engineering Framework (ICEF) prototype developed for RT30/31, and generate some real world data and feedback expressing its effectiveness as a systems engineering tool.

In this final stage of development, the synergy across RT30 and RT31 came to a pinnacle, and to provide the most value for research sponsors, the RT30a and RT31a components of ICEF sit on top of a common architecture framework. This framework now allows for the addition of new domains to facilitate the development of a CONOPS across a wide variety of systems. While the effort was based on a generic landscape and soldier domain, the ICEF architecture now allows a development team to create and utilize other domains – such as an urban warfare domain, a jungle warfare domain, or even a domain that represents a military installation. *As an example of this, a new research team member, a Stevens Institute sophomore with little coding background, has added a Healthcare domain using the ICEF environment and is modeling a hospital operating room and emergency room to investigate potential improvements in operating procedures. This same student is also doing the same for a dental office.*

This research has demonstrated that it is possible to utilize the strength of a 3D game development environment to create a graphical CONOPS tool that is easy enough for a non-technical system stakeholder to use, but also useful enough to provide value to system engineers and other development personnel. Feedback and data collected through interaction with ICEF's user community has shown that this alternative method of operational concept modeling is applicable to the current challenges faced by the DoD development and acquisitions community. Additionally, the work conducted under RT30/31 has served as a basis for the development of a novel concept engineering assessment model with complimentary CONOPS metrics. While not part of this specific research task, controlled experiments using ICEF have demonstrated the effectiveness of virtual, immersive, gaming environments to improve the shared mental model, and the quality of the developed CONOPS by individuals.

Finally, the research team was able to address an issue that the Model Based Systems Engineering community has consistently identified as an inhibitor of moving away from the traditional approach of systems engineering. Without full lifecycle support for MBSE methods and tools, systems engineers will continue to rely on the slow and cumbersome document driven engineering paradigm. Through development and testing of ICEF, the research team was able demonstrate that a CONOPS can be created following a model-driven process, and the output of such a process can be exported to an XML file and imported into a SysML tool. This is significant in that the CONOPS can now be a model-based basis of operational architecture.

APPENDICES

APPENDIX A: REFERENCES

Cloutier, R., Mostashari, A., McComb, S., Deshmukh, A., Wade, J., Kennedy, D., . . . Carrigy, A. (2010). Investigation of a Graphical CONOPS Development Environment for Agile Systems Engineering: Systems Engineering Research Center.

Cooper, G. F., & Yoo, C. (1999). *Causal discovery from a mixture of experimental and observational data*. Paper presented at the Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence, Stockholm, Sweden.

Esafahbod, Behnam, Integrated Concept Engineering Framework – a Visually-Rich Stakeholder-Centric System Design Software, Master's Thesis, Stevens Institute of Technology, May 2013.

Fenton, N., & Neil, M. (2012). *Risk Assessment and Decision Analysis With Bayesian Networks*: Taylor & Francis.

Fletcher, D. (2012). *A Comparative Analysis on Development Methods of Traditional and Graphical CONOPS*. Masters, Stevens Institute of Technology, Hoboken, NJ.

IEEE. (1998). IEEE Guide for Information Technology , System Definition , Concept of Operations (ConOps) Document. In I. C. S. Software Engineering Standards Committee of the (Ed.), (Vol. 1362-1998 (R2007)).

Kendall, T., & Hutchins, S. G. (2009). *Use of a team collaboration model to identify candidate knowledge management and collaboration technologies*. Paper presented at the Proceedings of the 9th Bi-annual international conference on Naturalistic Decision Making.

Korfiatis, P. (2013). *Development of a Virtual Concept Engineering Process to Extend Model-Based Systems Engineering*. PhD Dissertation, Stevens Institute of Technology, Hoboken, NJ.

Kruschke, J. (2010). *Doing Bayesian Data Analysis: A Tutorial Introduction with R*: Elsevier Science.

Kruschke, J. K. (2010). Bayesian data analysis. *Wiley Interdisciplinary Reviews: Cognitive Science, 1*(5), 658-676. doi: 10.1002/wcs.72

Kruschke, J. K. (2010). What to believe: Bayesian methods for data analysis. *Trends in cognitive sciences, 14*(7), 293-300.

Letsky, M., Warner, N., Fiore, S. M., Rosen, M., & Salas, E. (2007, June 19-21). *Macrocognition in complex team problem solving*. Paper presented at the International Command and Control Research and Technology Symposium, Newport, RI.

Linebarger, J., Scholand, A., Ehlen, M., & Procopio, M. (2005). *Benefits of synchronous collaboration support for an application-centered analysis team working on complex problems: a case study*. Paper presented at the GROUP '05:

Proceedings of the 2005 international ACM SIGGROUP conference on Supporting group work, Sanibel Island, Florida, USA.

Marek, J. D., & van der Gaag, L. C. (2000). Building Probabilistic Networks: 'Where Do the Numbers Come From?' Guest Editors' Introduction. *IEEE Transactions on Knowledge and Data Engineering, 12*(4), 481-486.

Masson, M. J. (2011). A tutorial on a practical Bayesian alternative to null-hypothesis significance testing. *Behavior Research Methods, 43*(3), 679-690. doi: 10.3758/s13428-010-0049-5

McComb, S. A. (2007). Mental model convergence: The shift from being an individual to being a team member. *Research in Multi Level Issues, 6*, 95-147.

Mostashari, A., McComb, S. A., Kennedy, D. M., Cloutier, R., & Korfiatis, P. (2011). Developing a Stakeholder-Assisted Agile CONOPS Development Process. *Systems Engineering, 15*(1), 1-13. doi: 10.1002/sys.20190

Noble, D., & Kirzl, J. (2003). *Objective Metrics for Evaluation of Collaborating Teams*. Paper presented at the Command and Control Research Technology Symposium, Washington, D.C.

Noble, D. F. (2004). Understanding and Applying the Cognitive Foundations of Effective Teamwork: Office of Naval Research.

Peters, R. D. Ideal Lift Kinematics - Complete Equations for Plotting Optimum Motion. *www.peters-research.com*. Retrieved from http://www.peters-research.com/index.php?option=com_content&view=article&id=61%3Aideal-lift-kinematics&catid=3%3Apapers&Itemid=1

Roberts, N., & Edson, R. (2008, October 21). *System Concept of Operations: Standards, Practices and Reality*. Paper presented at the NDIA 11th Annual Systems Engineering Conference, San Diego, CA.

Saldana, A. (2012). *Do Graphical CONOPS Help to Better Identify Integration and Testing Points, Planning, Estimation, and Overall Concept of Scenario?* Masters, Stevens Institute of Technology, Hoboken, NJ.

Uusitalo, L. (2007). Advantages and challenges of Bayesian networks in environmental modelling. *Ecological Modelling, 203*(3–4), 312-318. doi: http://dx.doi.org/10.1016/j.ecolmodel.2006.11.033

Wagenmakers, E.-J., Lodewyckx, T., Kuriyal, H., & Grasman, R. (2010). Bayesian hypothesis testing for psychologists: A tutorial on the Savage–Dickey method. *Cognitive Psychology, 60*(3), 158-189. doi: http://dx.doi.org/10.1016/j.cogpsych.2009.12.001

Warner, N., Letsky, M., & Cowen, M. (2005). Cognitive Model of Team Collaboration: Macro-Cognitive Focus. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting, 49*(3), 269-273. doi: 10.1177/154193120504900312

APPENDIX B: UNITY DEVELOPMENT GUIDELINE

Below are some guidelines unique to the Unity development environment. They are in essence, lessons learned for the developers.

B1 GENERAL RULES

A. The application must work, at all times. This means that it must be fully functional, with no red errors, and no yellow errors upon activation and operation. While yellow errors do not prevent making an executable, it's ADVISED AND PREFERABLE to eliminate them before making one. It eliminates phone-calls and questions later, too.
    a. It's okay to have less-than-perfect designs and/or algorithms.

B. We have legacy code here and there.
    a. DO NOT break code that is working.
    b. Update legacy code as needed.
    c. Coordinate with other team if you like to rewrite any of them.

C. Unity does not benefit from namespaces.
    a. Use meaningful and unique names for files/classes.

B2 PROJECT STRUCTURE

A. Every file should belong to a **module**.

B. Module names are UpperCamelCase. (except the prefixes that are described below)

C. Keep module names short, but descriptive and unique.

D. There a 4 types of modules:
    a. *Rooms. "Rxy AbcDef"*
        i. "xy" == "00" for the "Start" room
        ii. Each section of the application gets its own "x"
        Example: in ICEF, "Developer" section has "x" == "1", and "Scenario Authoring" section has "x" == "2"
    b. *Shared Libraries, "S AbcDef"*
    c. *Temporary, "T AbcDef"*
    d. *Legacy, "X AbcDef"*

E. Names of the folders inside the modules can start with an underscore ("_") to keep them at the top of the list of files/folder. (It's up to the module owner though)

B3 Files/Classes

A. In Unity (actually .Net), each C#/JS includes one class with the same name.
B. Names of files/classes must be UpperCamelCase.
C. **Do not put any files in the project root.**
D. It's better to not leave the names of files as "New ..."
    a. It's okay to do so as long as they are in a folder with a descriptive name.

B4 Steps to Take BEFORE Committing

A. **First, test your own work as it is.**
B. **Update your copy of the project from the main repository.**
    a. In case of conflicts:
        i. **Never "ignore the change from server"!**
        ii. **Always "merge" your work**
        iii. **If the automated merge doesn't work, do it manually**
    b. HINT: You shouldn't copy (overwrite) files after updating from server.
C. **Test it again!**
    a. Test your own work.
    b. If you've done a *manual merge*, test the general stability of the project.
D. **Compare *modified* files** and make sure there are no unintentional changes.
E. Make sure you are **not accidentally adding/removing** any files.

B5 Rules for Committing

A. Keep commits as small as possible (in number of affected files).
B. Commit as soon as possible.
    a. Every little change deserves its own commit, as long as it doesn't sacrifice the stability of the project.
C. *Commit message* must describe **all the major changes** in the commit.
D. If you need to reorganize some files/modules, do it in a completely separate commit.
    a. **Try to not "move" and "edit" in one commit.** Otherwise "automated merge" might not work for other teammates.
E. If you have to commit some unstable code, mention that in the commit message (i.e. add "(unstable)" to the beginning or ending of the message).

B6 Fixing a Broken State

A. The project is "broken" when any one or more of the following occurs:

  a. There is any compile-time error (red) or warning (yellow).
  b. There is any run-time error or warning, during *default* use-cases.
  c. Whenever any of the above rules for folders (modules/sub-modules) and files (classes) isn't followed.

B. When to fix it:
  a. **Always** fix a broken state before implementing/committing any new code.
  b. If *you* are responsible for the error/warning (meaning that the project has been broken by one of *your* commits), *you* should fix it **ASAP**!
    i. If you are *aware* of the problem, but cannot fix it right-away or by yourself alone, please write to the program manager **ASAP**!

C. How to fix it:
  a. Always fix a broken state in separate commits (not including any new feature, etc.)
  b. Use a reasonable solution to fix the problem.
    i. If there's a rush, you can disable (i.e. by marking as comment) some part of the code to fix the errors/warnings.
    ii. But do not disable a basic/required feature, just to get rid of the errors/warnings!
    iii. Find a smart solution using any available feature.
  c. Describe the problem and your solution in the commit message.

D. If you've notified the program manager about the problem in the past, remember to notify them when you've fixed it.

B7 NEW DOMAIN INSTALLATION GUIDELINES

The ICEF system is flexible yet robust, and able to handle multiple domains. In the system, domains are seen as a combination of working areas, actors, actions and related 3D representations.

- Create a new folder using the new domain name under /Domains/
  A javascript file describing the domain name, actors, actions and 3D models is needed
  - Function local_init (): Initialize the domain when the system starts
  - Function create_domain(): Initialize the domain using included domain type and object types
  - Function create_action_types(): Declare all actions and links used in the domain

- Function LinkType(string actionName, string animationName, Arrary objectType, string description): Save the link type object to the database, initialize the action
- Function create_primitives(): Initialize actors with 3D model.
- Function Primitive(stirng name, sting modelName, ObjType objectType, Array domain, bool isPlaceholder) enables the ICEF system to use 3D models in ICEF folder
- Function create_sample_actors (): Provider sample actors for user to add into scene easily
- Function create_a_scenario(): Defines initial scene and scenario

- Create a new folder using the new domain name under the root folder. Put all domain-related files inside this folder; these include working area and terrain materials, scenes, 3D models and associated materials, graphical user interfaces, animation scripts, and domain-specific scripts. Using existing javascript and c# code is recommended.
  - Folder _3D_Controller includes the codes that will be needed in the environment for users to move and control in the 3D space
    - Any changes in these files will change the way users act in the environment
  - Folder _3D_View includes the codes that will be needed in the environment for users to create new scenes, edit or delete scenes, and modify certain properties of the scenario
    - Any changes in these files will change the way users interact with basic scene and scenario options.
  - Folder _GUI_Contorller has the codes needed for user to interact with the system using graphical user interface
    - Any changes in these files will change what users can see and the way they interact with the system.
  - Folder _GUI_View has the codes controlling the left, right and bottom control panel of the GUI on the screen
    - Any changes in these files will change the way user interact with the system.
  - Folder _Model has the codes controlling the function of calling other codes
    - These codes are the hub of scene and scenario activities
  - Folder Resource has all the 3D models used in the domain[3]
    - All models used in the domain must also declared in the file inside /Domain/ folder

[3] 3D models inside Unity are global to all scripts, putting 3D models only in the Resource folder helps developers managing these files

- Create a new folder named "DomainName Selection", a javascript file used to initialize the database. This file also defines the dialog for users to choose from existing scenarios. Editing based on current MenuOpenScenario.js file is recommended.

B8 Unity Matlab Interface through TCP/IP Channel

IP – Internal system IP used here to communicate within the host computer
Default – 127.0.0.1
Port – Single port for both applications to communicate over
Range – 1024 – 49151

Matlab Portion:

```
IPVar = '127.0.0.1'
portVar = '11290'

%Create variable (TCPLine) containing TCP/IP data connection, assign it an IP and port
TCPLine = tcpip(IP,port)

%Open TCPIP Line
fopen(TCPLine)

%send unity the string "Initialized" to show matlab is initialized and ready to run
program
fprintf(TCPLine, 'Initialized');

%Close TCP Line and clean up variable
fclose(TCPLine);
delete(TCPLine);

%Return number of bytes in the buffer of TCPLine
get(TCPLine, 'BytesAvailable')

%Reads 1 byte from the TCPLine
fread(TCPLine,1)

%Sends string (tcpPacket) over TCP/IP connection
fprintf(TCPLine,tcpPacket);

%Waits until data is received, then stores string in InputBuffer
```

```
gotData=0;
inputBuffer=[];
while (gotData == 0)
pause(.1);      %pauses .1 seconds to reduce processor load
while (get(TCPLine, 'BytesAvailable') > 0)
inputBuffer = [inputBuffer char(fread(TCPLine,1))];
gotData = 1;
end
end
```

Unity Portion

```
//Open Matlab

//create variable matlab
matlab = new System.Diagnostics.Process ();

//Starts matlab minimized if in unity editor, hidden if in exe
#if UNITY_EDITOR
        matlab.StartInfo.WindowStyle = ProcessWindowStyle.Minimized;
#else
        matlab.StartInfo.WindowStyle = ProcessWindowStyle.Hidden;
#endif

//Gets path of current directory
string path = System.IO.Directory.GetCurrentDirectory ();

//creates path to deploy folder
path = Path.Combine (path, Deploy.GetPath (""));

//sets working directory for matlab, this is where the function to be called must be
stored
matlab.StartInfo.WorkingDirectory = path;

//gives command for matlab to run the vehicle simulation function, with argument port,
and turn off the splash and desktop, making it more transparent to end user
string functionInput = "-r VehicleSimulation(" + port.ToString () + ")" + " -nosplash -
nodesktop";
matlab.StartInfo.Arguments = functionInput;

//sets the matlab variable to open matlab
matlab.StartInfo.FileName = "matlab.exe";

//Starts matlab with given criteria
```

```
matlab.Start ();

//Initializations--
        private bool mRunning;
        Thread mThread;
        TcpListener tcp_Listener = null;
        int port  = 11290;
        IPAddress localAddr;
        TcpClient client;
        NetworkStream ns;

        ASCIIEncoding encoder;
        byte[] OutputBuffer;

//sets up a parallel thread (receive data function) to receive data in
        mRunning = true;
        ThreadStart ts = new ThreadStart (receiveData);
        mThread = new Thread (ts);
        mThread.Start ();

void receiveData ()
{
while (mRunning) {
        try {
                localAddr = IPAddress.Parse ("127.0.0.1");
                tcp_Listener = new TcpListener (localAddr, port);
                tcp_Listener.Start ();
                client = tcp_Listener.AcceptTcpClient ();
ns = client.GetStream ();
                Byte[] bytes = new Byte[256];
String data = null;
                while ((i = ns.Read (bytes, 0, bytes.Length)) != 0) {
                        // Translate data bytes to a ASCII string.
                        data = System.Text.Encoding.ASCII.GetString (bytes, 0, i);
        } catch (ThreadAbortException e) {
                //UnityEngine.Debug.Log ("exception - possibly client disconnected: " +
e);
        } catch (Exception e) {
                //UnityEngine.Debug.Log ("Exception thrown: " + e);
        }
}
stopListening ();  //closes listening thread
}
```

```
void OnApplicationQuit ()    //executes if application quits, prevents ghost matlab
instances
{
        stopListening ();      //quits listening server
        matlab.Kill ();                    //exits matlab
}

// create a new encoder to encode the characters to bytes
encoder = new ASCIIEncoding ();

// convert 'Output' to bytes and store in output buffer
OutputBuffer = encoder.GetBytes (Output);

//write OutputBuffer to TCP/IP Line
ns.Write (OutputBuffer, 0, OutputBuffer.Length);
```

Appendix C: CSharp_Code_AtRiskInterface

```
////////////////////////////////////////////
// This C#-based file calls the @Risk SDK functions
// (which calculate probability distributions) and returns
// the computation solutions to Unity.
////////////////////////////////////////////

\Program.cs
using System;
using System.Collections.Generic;
using System.Text;
using System.IO;
using System.Drawing;
using System.Diagnostics;
using LitJson;
using Palisade.RdkPia;
using System.Runtime.InteropServices;

namespace AtRiskInterface
{
  class Program
  {

    [DllImport("user32.dll")]
    public static extern IntPtr FindWindow(string lpClassName, string
lpWindowName);

    [DllImport("user32.dll")]
    public static extern bool ShowWindow(IntPtr hWnd, int nCmdShow);


    static void Main(string[] args)
    {
      Console.Title = "AtRiskInterface";
      IntPtr hwnd = FindWindow(null, "AtRiskInterface");
      ShowWindow(hwnd, 0);

      Console.InputEncoding = System.Text.Encoding.UTF8;
      Console.OutputEncoding = System.Text.Encoding.UTF8;

      string inputJson = Console.In.ReadToEnd();

      JsonData inputData = null;
```

```

            try
            {
                inputData = LitJson.JsonMapper.ToObject(inputJson);
            }
            catch (JsonException ex)
            {
                Console.Error.WriteLine("ERROR Parsing Input:" + ex.ToString());
            }

            JsonData outputData = CallAtRisk(inputData);
            string outputJson = null;

            try
            {
                outputJson = LitJson.JsonMapper.ToJson(outputData);
            }
            catch (JsonException ex)
            {
                Console.Error.WriteLine("ERROR Generating Output:" + ex.ToString());
            }
            Console.Out.Write(outputJson);
            Console.Out.Close();
        }

        public static JsonData CallAtRisk(JsonData inputData)
        {
            String processId = Process.GetCurrentProcess().Id.ToString();
            String tempImageFile = Environment.GetEnvironmentVariable("UserProfile") +
"\\Desktop\\AtRiskImage-" + processId + ".jpg";

            RDKApplication myRdkApp = new RDKApplication();
            myRdkApp.Initialize(false);
            myRdkApp.GraphDefaults.DestinationFile = tempImageFile;
            myRdkApp.GraphDefaults.Destination = RDKGraphDestination.RDKJPGFile;

            Console.Error.WriteLine("InputDist: " + inputData["Input"].ToString());
            RDKInput myIn = myRdkApp.Inputs.Add(inputData["Name"].ToString(),
inputData["Input"].ToString());


myRdkApp.SimSettings.set_NumIterations(int.Parse(inputData["NumIterations"].ToSt
ring()));
```

```

myRdkApp.SimSettings.set_NumSimulations(int.Parse(inputData["NumSimulations"].
ToString()));

            myRdkApp.Simulate();
            //var pic =
myIn.Results[1].Graph(RDKResultCurveType.RDKResultCurveTypeHistogram,myIn.R
esults[1]);


            //Image myImage = new Bitmap(Image.FromFile(tempImageFile), new
Size(220, 220));
            //MemoryStream myImageMemStream = new MemoryStream();
            //myImage.Save(myImageMemStream,
System.Drawing.Imaging.ImageFormat.Jpeg);
            //byte[] imgByteArray = myImageMemStream.GetBuffer();


            JsonData output = new JsonData();
            output["min"] = myIn.Results[1].Statistics.Minimum;
            output["mean"] = myIn.Results[1].Statistics.Mean;
            output["max"] = myIn.Results[1].Statistics.Maximum;
            //output["image"] = Convert.ToBase64String(imgByteArray);

            myRdkApp.Free();

            return output;
        }
    }
}



////////////////////////////////
\Properties\AssemblyInfo.cs
using System.Reflection;
using System.Runtime.CompilerServices;
using System.Runtime.InteropServices;

// General Information about an assembly is controlled through the following
// set of attributes. Change these attribute values to modify the information
// associated with an assembly.
[assembly: AssemblyTitle("AtRiskInterface")]
```

```
[assembly: AssemblyDescription("")]
[assembly: AssemblyConfiguration("")]
[assembly: AssemblyCompany("")]
[assembly: AssemblyProduct("AtRiskInterface")]
[assembly: AssemblyCopyright("Copyright ©  2012")]
[assembly: AssemblyTrademark("")]
[assembly: AssemblyCulture("")]

// Setting ComVisible to false makes the types in this assembly not visible
// to COM components.  If you need to access a type in this assembly from
// COM, set the ComVisible attribute to true on that type.
[assembly: ComVisible(false)]

// The following GUID is for the ID of the typelib if this project is exposed to COM
[assembly: Guid("63f66463-85c2-4d68-aca5-a56a42bb74b6")]

// Version information for an assembly consists of the following four values:
//
//      Major Version
//      Minor Version
//      Build Number
//      Revision
//
// You can specify all the values or you can default the Build and Revision Numbers
// by using the '*' as shown below:
// [assembly: AssemblyVersion("1.0.*")]
[assembly: AssemblyVersion("1.0.0.0")]
[assembly: AssemblyFileVersion("1.0.0.0")]
```

APPENDIX D: CSHARP_CODE_CONOPS_MICROSOFTINTERFACE

```
/////////////////////////////////////////////////////
// This C#-based file launches Excel and // computes within Excel, and returns the
solution of the
// computations to Unity. It also generates a MS-Word
// document in which the computational solutions can
// be saved.
///////////////////////////////////////////////////

\ExcelFunctions.cs
using System;
using Excel = Microsoft.Office.Interop.Excel;

namespace CONOPS_MicrosoftInterface
{
  public class ExcelFunctions
  {
    Excel.Application oXL;

    //Function call of Excel Skew
    public double Skew(double[] data)
    {
      oXL = new Excel.Application();
      double result = oXL.WorksheetFunction.Skew(data);
      oXL.Quit();
      System.Runtime.InteropServices.Marshal.ReleaseComObject(oXL);
      GC.Collect();
      return result;
    }

    //Function call of Excecl Kurtosis
    public double Kurt(double[] data)
    {
      oXL = new Excel.Application();
      double result = oXL.WorksheetFunction.Kurt(data);
      oXL.Quit();
      System.Runtime.InteropServices.Marshal.ReleaseComObject(oXL);
      GC.Collect();
      return result;
    }

    //Function call of Excel Standard Deviation based on the entire population
    public double Std_Dev(double[] data)
```

```
        {
          oXL = new Excel.Application();
          double result = oXL.WorksheetFunction.StDev_P(data);
          oXL.Quit();
          System.Runtime.InteropServices.Marshal.ReleaseComObject(oXL);
          GC.Collect();
          return result;
        }

        //Function call of Excel Average, calculate the mean
        public double Mean(double[] data)
        {
          oXL = new Excel.Application();
          double result = oXL.WorksheetFunction.Average(data);
          oXL.Quit();
          System.Runtime.InteropServices.Marshal.ReleaseComObject(oXL);
          GC.Collect();
          return result;
        }

        //End the Excel application if it didn't quit successfully.
        public void EndExcelInstance()
        {
          oXL.Quit();
          System.Runtime.InteropServices.Marshal.ReleaseComObject(oXL);
          GC.Collect();
        }
      }
    }



////////////////////////////////
\FunctionHub.cs
using System;
using System.IO.Pipes;
using Excel = Microsoft.Office.Interop.Excel;
using AdobePDF = AdobePDFMakerX;

namespace CONOPS_MicrosoftInterface
{
  public class FunctionHub
  {
        NamedPipeServerStream pipeServer;
        ExcelFunctions exF;

        public void RunPipe()
        {
            while (true)
            {
                LaunchServer();
            }
        }

        public void LaunchServer()
        {
            //initialize the pipe.
            try
            {
                pipeServer = new NamedPipeServerStream("CONOPSInterfacePipe",
PipeDirection.InOut, 1);
                pipeServer.WaitForConnection();
            }
            catch
            {
                Environment.Exit(1);
            }
            try
            {
                // Attach pipe as stream.
                StreamString ss = new StreamString(pipeServer);

                // Read the request from the client.
                string request = ss.ReadString();
                exF = new ExcelFunctions();
                string firstData;
                int numNumber;
                double[] num;
                string numReq;
                if (request == "ExportWord")
                {
                    //Read string from pipe and export to word document.
                    firstData = ss.ReadString();
                    WordFunctions wf = new WordFunctions();
                    //Show only the save dialog of Word, hide the Word Window.
                    wf.ExportToWordWithoutShowing(firstData);
            }
            else if (request == "OpenInWord")
            {
               firstData = ss.ReadString();
               WordFunctions wf = new WordFunctions();
               wf.ExportToWord(firstData);
            }
            else if (request == "ExportPdf")
            {
               //implement pdf converter here.
            }
            else
            {
               //read data from pipe, number of the data set.
               numReq = ss.ReadString();
               numNumber = Int16.Parse(numReq);
               num = new double[numNumber];

               //read all data to process from the pipe.
               for (int i = 0; i < numNumber; i++)
               {
                  num[i] = double.Parse(ss.ReadString());
               }
               double result = 0.0;
               if (request == "SKEW")
               {
                  //process data with Skew function
                  result = exF.Skew(num);
                  ss.WriteString(result.ToString());
                  exF.EndExcelInstance();
               }
               else if (request == "KURTOSIS")
               {
                  //process data with Kurtosis funciton
                  result = exF.Kurt(num);
                  ss.WriteString(result.ToString());
                  exF.EndExcelInstance();
               }
               else if (request == "StDev_P")
               {
                  //process data with StandardDeviate function (based on entire population)
                  result = exF.Std_Dev(num);
                  ss.WriteString(result.ToString());
```

```
                    exF.EndExcelInstance();
                }
                else if (request == "Mean")
                {
                    //compute the mean of data set.
                    result = exF.Std_Dev(num);
                    ss.WriteString(result.ToString());
                    exF.EndExcelInstance();
                }
                else if (request == "All Stats")
                {
                    //process data with all four functions.
                    result = exF.Skew(num);
                    ss.WriteString(result.ToString());
                    result = exF.Kurt(num);
                    ss.WriteString(result.ToString());
                    result = exF.Std_Dev(num);
                    ss.WriteString(result.ToString());
                    result = exF.Mean(num);
                    ss.WriteString(result.ToString());
                    exF.EndExcelInstance();
                }
              }
            }
            catch
            {
              Environment.Exit(1);
            }
            pipeServer.Close();
        }
    }
}



////////////////////////////////
\Program.cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Forms;

namespace CONOPS_MicrosoftInterface
```

```
{
   static class Program
   {
      /// <summary>
      /// The main entry point for the application.
      /// </summary>
      [STAThread]
      static void Main()
      {
         FunctionHub FH = new FunctionHub();
         FH.RunPipe();
      }
   }
}



////////////////////////////////
\StreamString.cs
using System;
using System.IO;
using System.Text;

namespace CONOPS_MicrosoftInterface
{
   public class StreamString
   {
      private Stream ioStream;
      private UnicodeEncoding streamEncoding;

      public StreamString(Stream ioStream)
      {
         this.ioStream = ioStream;
         streamEncoding = new UnicodeEncoding();
      }

      public string ReadString()
      {
         int len = 0;

         len = ioStream.ReadByte() * 256;
         len += ioStream.ReadByte();
         byte[] inBuffer = new byte[len];
```

```
            ioStream.Read(inBuffer, 0, len);
            return streamEncoding.GetString(inBuffer);
        }

        public int WriteString(string outString)
        {
            byte[] outBuffer = streamEncoding.GetBytes(outString);
            int len = outBuffer.Length;
            if (len > UInt16.MaxValue)
            {
                len = (int)UInt16.MaxValue;
            }
            ioStream.WriteByte((byte)(len / 256));
            ioStream.WriteByte((byte)(len & 255));
            ioStream.Write(outBuffer, 0, len);
            ioStream.Flush();
            return outBuffer.Length + 2;
        }
    }
}



////////////////////////////////
\WordFunctions.cs
using System;
using Word = Microsoft.Office.Interop.Word;

namespace CONOPS_MicrosoftInterface
{
    public class WordFunctions
    {
        object oMissing = System.Reflection.Missing.Value;
        object oEndOfDoc = "\\endofdoc";
        Word._Application oWord;
        Word._Document oDoc;

        //Launch Word application and save data to Word document without showing the
Word Window.
        public void ExportToWordWithoutShowing(string data)
        {
            oWord = new Word.Application();
            oWord.Visible = false;
            oDoc = oWord.Documents.Add(ref oMissing, ref oMissing, ref oMissing, ref
oMissing);

            Word.Paragraph oPara1;
            oPara1 = oDoc.Content.Paragraphs.Add(ref oMissing);
            oPara1.Range.Text = String.Format("{0}\n{1}", data, DateTime.Now);
            oPara1.Format.SpaceAfter = 24;
            oPara1.Range.InsertParagraphAfter();
            oDoc.Save();
            oDoc.Close();
            oWord.Quit();
            System.Runtime.InteropServices.Marshal.ReleaseComObject(oWord);
            GC.Collect();
        }

        //Launch Word application and save data to Word document.
        public void ExportToWord(string data)
        {
            oWord = new Word.Application();
            oWord.Visible = true;
            oDoc = oWord.Documents.Add(ref oMissing, ref oMissing, ref oMissing, ref
oMissing);

            Word.Paragraph oPara1;
            oPara1 = oDoc.Content.Paragraphs.Add(ref oMissing);
            oPara1.Range.Text = String.Format("{0}\n{1}", data, DateTime.Now);
            oPara1.Format.SpaceAfter = 24;
            oPara1.Range.InsertParagraphAfter();
            //oDoc.Save();
            //oDoc.Close();
            //oWord.Quit();
            System.Runtime.InteropServices.Marshal.ReleaseComObject(oWord);
            GC.Collect();
        }
    }
}



////////////////////////////////
\Properties\AssemblyInfo.cs
using System.Reflection;
using System.Runtime.CompilerServices;
using System.Runtime.InteropServices;

// General Information about an assembly is controlled through the following
// set of attributes. Change these attribute values to modify the information
// associated with an assembly.
[assembly: AssemblyTitle("CONOPS_MicrosoftInterface")]
[assembly: AssemblyDescription("")]
[assembly: AssemblyConfiguration("")]
[assembly: AssemblyCompany("")]
[assembly: AssemblyProduct("CONOPS_MicrosoftInterface")]
[assembly: AssemblyCopyright("Copyright ©  2011")]
[assembly: AssemblyTrademark("")]
[assembly: AssemblyCulture("")]

// Setting ComVisible to false makes the types in this assembly not visible
// to COM components.  If you need to access a type in this assembly from
// COM, set the ComVisible attribute to true on that type.
[assembly: ComVisible(false)]

// The following GUID is for the ID of the typelib if this project is exposed to COM
[assembly: Guid("1f04075c-a909-49b3-aadc-1d95aa243bce")]

// Version information for an assembly consists of the following four values:
//
//      Major Version
//      Minor Version
//      Build Number
//      Revision
//
// You can specify all the values or you can default the Build and Revision Numbers
// by using the '*' as shown below:
// [assembly: AssemblyVersion("1.0.*")]
[assembly: AssemblyVersion("1.0.0.0")]
[assembly: AssemblyFileVersion("1.0.0.0")]



/////////////////////////////////
\Properties\Resources.Designer.cs
//------------------------------------------------------------------------------
// <auto-generated>
//     This code was generated by a tool.
//     Runtime Version:4.0.30319.235
```

```
//
//     Changes to this file may cause incorrect behavior and will be lost if
//     the code is regenerated.
// </auto-generated>
//------------------------------------------------------------------------------

namespace CONOPS_MicrosoftInterface.Properties
{


    /// <summary>
    ///   A strongly-typed resource class, for looking up localized strings, etc.
    /// </summary>
    // This class was auto-generated by the StronglyTypedResourceBuilder
    // class via a tool like ResGen or Visual Studio.
    // To add or remove a member, edit your .ResX file then rerun ResGen
    // with the /str option, or rebuild your VS project.

[global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Resources.Tools.
StronglyTypedResourceBuilder", "4.0.0.0")]
    [global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
    [global::System.Runtime.CompilerServices.CompilerGeneratedAttribute()]
    internal class Resources
    {

        private static global::System.Resources.ResourceManager resourceMan;

        private static global::System.Globalization.CultureInfo resourceCulture;


[global::System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Perfor
mance", "CA1811:AvoidUncalledPrivateCode")]
        internal Resources()
        {
        }

        /// <summary>
        ///   Returns the cached ResourceManager instance used by this class.
        /// </summary>

[global::System.ComponentModel.EditorBrowsableAttribute(global::System.Componen
tModel.EditorBrowsableState.Advanced)]
        internal static global::System.Resources.ResourceManager ResourceManager
        {
          get
          {
            if ((resourceMan == null))
            {
              global::System.Resources.ResourceManager temp = new
global::System.Resources.ResourceManager("CONOPS_MicrosoftInterface.Properties.
Resources", typeof(Resources).Assembly);
              resourceMan = temp;
            }
            return resourceMan;
          }
        }

        /// <summary>
        ///   Overrides the current thread's CurrentUICulture property for all
        ///   resource lookups using this strongly typed resource class.
        /// </summary>

[global::System.ComponentModel.EditorBrowsableAttribute(global::System.Componen
tModel.EditorBrowsableState.Advanced)]
        internal static global::System.Globalization.CultureInfo Culture
        {
          get
          {
            return resourceCulture;
          }
          set
          {
            resourceCulture = value;
          }
        }
    }
}



/////////////////////////////////
\Properties\Settings.Designer.cs
//------------------------------------------------------------------------------
// <auto-generated>
//     This code was generated by a tool.
//     Runtime Version:4.0.30319.235
```

```
//
//     Changes to this file may cause incorrect behavior and will be lost if
//     the code is regenerated.
// </auto-generated>
//------------------------------------------------------------------------------

namespace CONOPS_MicrosoftInterface.Properties
{


    [global::System.Runtime.CompilerServices.CompilerGeneratedAttribute()]

[global::System.CodeDom.Compiler.GeneratedCodeAttribute("Microsoft.VisualStudio.E
ditors.SettingsDesigner.SettingsSingleFileGenerator", "10.0.0.0")]
    internal sealed partial class Settings :
global::System.Configuration.ApplicationSettingsBase
    {

        private static Settings defaultInstance =
((Settings)(global::System.Configuration.ApplicationSettingsBase.Synchronized(new
Settings())));

        public static Settings Default
        {
            get
            {
                return defaultInstance;
            }
        }
    }
}
```

APPENDIX E: CSHARP_CODE_EXCELREADER

```
/////////////////////////////////////////////////
// This C#-based application initializes a pipe to
// communicate with Unity, in order to read an
// Excel file, and forward the data to Unity
////////////////////////////////////////////////

\Form1.cs
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Diagnostics;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using Microsoft.Office.Interop.Excel;
using Microsoft.Office.Core;
using System.IO.Pipes;

namespace CONOPS_Lobby_ExcelReader
{
   public partial class Form1 : Form
   {
      NamedPipeServerStream pipeServer;
      public Form1()
      {
         InitializeComponent();
         try
         {
            System.IO.FileInfo fi = new
System.IO.FileInfo(@"Deploy\VehiclesAllocationData.xlsx");
            if (fi.Exists)
               RunPipe();
            else
            {
               MessageBox.Show("File does not found, please make sure the
VehiclesAllocationData.xlsx is in the deploy folder.");
               System.Environment.Exit(1);
            }
         }
         catch
```

```
            {
               MessageBox.Show("File does not found, please make sure the
VehiclesAllocationData.xlsx is in the deploy folder.");
               System.Environment.Exit(1);
            }

        }

        public string[] ReadExcel(int sheetNo)
        {
            string filename = new
System.IO.FileInfo(@"Deploy\VehiclesAllocationData.xlsx").FullName;
            string[] exceldata = new string[8];
            Microsoft.Office.Interop.Excel.Application ExcelObj = new
Microsoft.Office.Interop.Excel.Application();
            Microsoft.Office.Interop.Excel.Workbook theWorkbook =
ExcelObj.Workbooks.Open(
              filename, 0, true, 5,
               "", "", true, Microsoft.Office.Interop.Excel.XlPlatform.xlWindows, "\t", false,
false,
               0, true);
            Microsoft.Office.Interop.Excel.Sheets sheets = theWorkbook.Worksheets;
            Microsoft.Office.Interop.Excel.Worksheet worksheet =
(Microsoft.Office.Interop.Excel.Worksheet)sheets.get_Item(sheetNo);
            for (int i = 6; i <= 13; i++)
            {
               Microsoft.Office.Interop.Excel.Range range = worksheet.get_Range("A" +
i.ToString(), "B" + i.ToString());
               System.Array myvalues = (System.Array)range.Cells.Value;
               string[] strArray = ConvertToStringArray(myvalues);
               exceldata[i - 6] = strArray[1];
            }
            sheets = null;
            theWorkbook.Close(false);
            theWorkbook = null;
            worksheet = null;
            ExcelObj.Quit();
            NAR(ExcelObj);
            System.Runtime.InteropServices.Marshal.ReleaseComObject(ExcelObj);
            return exceldata;
        }

        string[] ConvertToStringArray(System.Array values)
        {
            string[] theArray = new string[values.Length];
            for (int i = 1; i <= values.Length; i++)
            {
                if (values.GetValue(1, i) == null)
                    theArray[i - 1] = "";
                else
                    theArray[i - 1] = (string)values.GetValue(1, i).ToString();
            }
            return theArray;
        }

        public void RunPipe()
        {
            LaunchServer();
        }

        public void LaunchServer()
        {
            //initialize the pipe.
            try
            {
                pipeServer = new NamedPipeServerStream("CONOPSExcelReaderPipe",
PipeDirection.InOut, 1);
                pipeServer.WaitForConnection();
                // Attach pipe as stream.
                StreamString ss = new StreamString(pipeServer);

                // Read the request from the client.
                string request = ss.ReadString();
                if (request == "ReadExcel")
                {
                    for (int j = 1; j < 6; j++)
                    {
                        string[] exceldata = new string[8];
                        exceldata = ReadExcel(j);
                        for (int i = 0; i < exceldata.Length; i++)
                            ss.WriteString(exceldata[i]);
                    }
                }
                pipeServer.Close();
            }
            catch
            {
                System.Environment.Exit(1);
            }
            GC.Collect();
        }

        private void NAR(object o)
        {
            try
            {
                System.Runtime.InteropServices.Marshal.ReleaseComObject(o);
            }
            catch { }
            finally
            {
                o = null;
            }
        }
    }
}



////////////////////////////////
\Form1.Designer.cs
namespace CONOPS_Lobby_ExcelReader
{
    partial class Form1
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be disposed;
otherwise, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.openFileDialog1 = new System.Windows.Forms.OpenFileDialog();
            this.SuspendLayout();
            //
            // openFileDialog1
            //
            this.openFileDialog1.FileName = "openFileDialog1";
            //
            // Form1
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(0, 0);
            this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.None;
            this.Name = "Form1";
            this.ShowInTaskbar = false;
            this.StartPosition = System.Windows.Forms.FormStartPosition.CenterParent;
            this.Text = "Form1";
            this.WindowState = System.Windows.Forms.FormWindowState.Minimized;
            this.ResumeLayout(false);

        }

        #endregion

        private System.Windows.Forms.OpenFileDialog openFileDialog1;
    }
}
```

```

/////////////////////////////////
\Program.cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Forms;

namespace CONOPS_Lobby_ExcelReader
{
   static class Program
   {
      /// <summary>
      /// The main entry point for the application.
      /// </summary>
      [STAThread]
      static void Main()
      {
         Application.EnableVisualStyles();
         Application.SetCompatibleTextRenderingDefault(false);
         Application.Run(new Form1());
         System.Environment.Exit(0);
      }
   }
}



/////////////////////////////////
\StreamString.cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;

namespace CONOPS_Lobby_ExcelReader
{
   class StreamString
   {
      private Stream ioStream;
      private UnicodeEncoding streamEncoding;

```

```
        public StreamString(Stream ioStream)
        {
            this.ioStream = ioStream;
            streamEncoding = new UnicodeEncoding();
        }

        public string ReadString()
        {
            int len = 0;

            len = ioStream.ReadByte() * 256;
            len += ioStream.ReadByte();
            byte[] inBuffer = new byte[len];
            ioStream.Read(inBuffer, 0, len);
            return streamEncoding.GetString(inBuffer);
        }

        public int WriteString(string outString)
        {
            byte[] outBuffer = streamEncoding.GetBytes(outString);
            int len = outBuffer.Length;
            if (len > UInt16.MaxValue)
            {
                len = (int)UInt16.MaxValue;
            }
            ioStream.WriteByte((byte)(len / 256));
            ioStream.WriteByte((byte)(len & 255));
            ioStream.Write(outBuffer, 0, len);
            ioStream.Flush();
            return outBuffer.Length + 2;
        }
    }
}
```

APPENDIX F: UNITY_CODE_MATLAB_INVOCATION_USED_IN_RT31

```
//////////////////////////////////////////////////////////////////////
// These are the Unity environment C# script functions which govern the
// setting up of the TCP/IP communication
// channel between Unity and MATLAB, which calculate the velocity,
// acceleration, fuel consumption, etc. for
// Vehicle Simulation.
///////////////////////////////////////////////////////////////////////

using UnityEngine;
using System.Text;
using System.Net;
using System.Net.Sockets;
using System.Diagnostics;
using System.Collections;
using System.IO;
using System.Threading;
using System;

public class MatlabImport : MonoBehaviour
{
	System.Diagnostics.Process matlab;
	private bool mRunning;
	Thread mThread;
	TcpListener tcp_Listener = null;
	int port  = 11290;
	IPAddress localAddr;
	TcpClient client;
	NetworkStream ns;

	private bool endCondition;
	private string[] parsedValues = { "0", "0", "0", "0", "0" };

	GameObject IsoCamera = null;
	GameObject VertCamera = null;
	GameObject FreeCamera = null;
	GameObject CameraAxes = null;

	GameObject Humvee = null;
	GameObject Car = null;
	GameObject jltv = null;
	GameObject M113 = null;
```

```
        GameObject matv01_veh = null;
        GameObject Mrap = null;
        GameObject stryker = null;

        private float startPos;
        public bool runningSim;
        public bool isPaused;
        public bool MatlabLoaded;
        public bool Nomatlab;

        public string PauseOrResume = "Run";
        public bool write2file;
        public string DataStorageFilenameString = "VehicleSimulationData";
        public bool dispGraphs;

        public float distanceValue = 1.0f;
        public float velocityValue = 80.0f;
        public float accelerationValue = 10.0f;

        public float distanceMin = 0.1f;
        public float distanceMax = 4.0f;
        public float velocityMin = 0.1f;
        public float velocityMax = 80.0f;
        public float accelerationMin = 0.1f;
        public float accelerationMax = 10.0f;

        public string[] vehicleNames = { "Stryker", "M-ATV", "MRAP", "M113",
"Humvee", "JLTV", "Car" };
        public float[,] vehicleProperties = { { 62.0f, 65.0f, 65.0f, 42.0f, 65.0f, 70.0f, 85.0f
}, { 3.0f, 1.829f, 3.0f, 3.0f, 3.0f, 3.0f, 3.5f }, { 0.9151107f, 0.6332513f, 0.9338072f,
0.6049597f, 0.6193988f, 0.6863496f, 0.6929922f } };
        public int activeVehicle = 6;
        public int numVehicles = 7;

        ASCIIEncoding encoder;
        byte[] OutputBuffer;

        void Start ()
        {
                Nomatlab = false;
                try {
                        OpenMatlab (port);
                }
```

```
			catch {
				Nomatlab = true;
			}
			MatlabLoaded = false;
			endCondition = false;
			dispGraphs = false;
			restartScene();

			Humvee = GameObject.Find ("Humvee");
			//Note for future development:  change this into an array of game objects
			Car = GameObject.Find ("PersonalCar");
			jltv = GameObject.Find ("jltv");
			M113 = GameObject.Find ("M113");
			matv01_veh = GameObject.Find ("matv01_veh");
			Mrap = GameObject.Find ("mrap");
			stryker = GameObject.Find ("stryker");

			IsoCamera = GameObject.Find ("Main Camera");
			VertCamera = GameObject.Find ("VertCamera");
			FreeCamera = GameObject.Find ("FreeLookCamera");
			CameraAxes = GameObject.Find ("CameraAxes");

			IsoCamera.camera.enabled = true;
			VertCamera.camera.enabled = false;
			FreeCamera.camera.enabled = false;

			updateVehicles ();

			//set up TCP/IP client
			mRunning = true;
			ThreadStart ts = new ThreadStart (receiveData);
			mThread = new Thread (ts);
			mThread.Start ();

			startPos = transform.position.x;
		}

		// Update is called once per frame
		void Update ()
		{
			float x = float.Parse (parsedValues[0]);
			transform.position = new Vector3 (startPos + x / 5.725f, 0, 3.785523f);
			//implement scale 1 unity unit=~5.725 ft
```

```
        }

        public void OpenMatlab (int port)
        {
                matlab = new System.Diagnostics.Process ();
                string path = System.IO.Directory.GetCurrentDirectory ();
                #if UNITY_EDITOR
                //path = Path.Combine(path, "Assets");
                matlab.StartInfo.WindowStyle = ProcessWindowStyle.Minimized;
                #else
                matlab.StartInfo.WindowStyle = ProcessWindowStyle.Hidden;
                #endif
                path = Path.Combine (path, Deploy.GetPath (""));
                string functionInput = "-r VehicleSimulation(" + port.ToString () + ")" + "
    -nosplash -nodesktop";
                matlab.StartInfo.WorkingDirectory = path;
                matlab.StartInfo.FileName = "matlab.exe";
                matlab.StartInfo.Arguments = functionInput;
                matlab.Start ();
        }

        void OnGUI ()
        {
                int Spacing = 5;
                int SimLabelWidth = 80;
                int SimLabelHeight = 40;
                int SimGUIWidth = 5 * Spacing + 4 * SimLabelWidth;
                int SimGUIHeight = 2 * SimLabelHeight + 3 * Spacing;

                int InitiateLabelWidth = 100;
                int InitiateScrollWidth = 170;
                int InitiateGUIWidth = 2 * InitiateScrollWidth + InitiateLabelWidth + 3 *
    Spacing;
                int InitiateGUIHeight = 175;

                if (Nomatlab) {
                        if (GUI.Button(new Rect(Screen.width/2-10/2,Screen.height/2-
    25,200,50),"Matlab Failed to Load\nClick to Exit")) {
                                Application.LoadLevel ("RoomLobby");
                                try{
                                        stopListening ();
                                }
                                catch{}
```

```
            }
        }

        else if (runningSim) {
            GUI.Box (new Rect (Screen.width / 2 - SimGUIWidth / 2, 30,
SimGUIWidth, SimGUIHeight), "");
            GUILayout.BeginArea (new Rect (Screen.width / 2 - SimGUIWidth
/ 2, 30, SimGUIWidth, SimGUIHeight));
            GUILayout.BeginVertical ();
            GUILayout.Space (Spacing);
            GUILayout.BeginHorizontal ();
            GUILayout.Space (Spacing);
            GUILayout.Box ("Time\n" + parsedValues[1] + " sec",
GUILayout.Width (SimLabelWidth), GUILayout.Height (SimLabelHeight));
            GUILayout.Box ("Distance\n" + parsedValues[2] + " mi",
GUILayout.Width (SimLabelWidth), GUILayout.Height (SimLabelHeight));
            GUILayout.Box ("Velocity\n" + parsedValues[3] + " mph",
GUILayout.Width (SimLabelWidth), GUILayout.Height (SimLabelHeight));
            GUILayout.Box ("Acceleration\n" + parsedValues[4] + "mph/s",
GUILayout.Width (SimLabelWidth), GUILayout.Height (SimLabelHeight));
            GUILayout.Space (Spacing);
            GUILayout.EndHorizontal ();
            GUILayout.BeginHorizontal ();
            GUILayout.Space (Spacing);
            if (GUILayout.Button ("Change\nPOV", GUILayout.Width
(SimLabelWidth), GUILayout.Height (SimLabelHeight))) {
                if (IsoCamera.camera.enabled) {
                    IsoCamera.camera.enabled = false;
                    VertCamera.camera.enabled = true;
                } else if (VertCamera.camera.enabled) {
                    VertCamera.camera.enabled = false;
                    FreeCamera.camera.enabled = true;
                } else {
                    FreeCamera.camera.enabled = false;
                    IsoCamera.camera.enabled = true;
                }
            }
            if (GUILayout.Button (PauseOrResume, GUILayout.Width
(SimLabelWidth), GUILayout.Height (SimLabelHeight))) {
                if (isPaused) {
                    sendResume ();
                } else {
                    sendPause ();
                }
            }
            if (GUILayout.Button ("Cancel\nSimulation", GUILayout.Width
(SimLabelWidth), GUILayout.Height (SimLabelHeight))) {
                sendRestart ();
            }
            /*if (GUILayout.Button ("Graphs\n(N/A yet)", GUILayout.Width
(SimLabelWidth), GUILayout.Height (SimLabelHeight))) {
                if (isGraph) {
                    //send resume command
                    sendGraphOff ();
                } else {
                    //send pause command
                    sendGraphOn ();
                }
            }*/
            if (GUILayout.Button ("Return\nto Lobby", GUILayout.Width
(SimLabelWidth), GUILayout.Height (SimLabelHeight))) {
                sendExit();
            }
            GUILayout.Space (Spacing);
            GUILayout.EndHorizontal ();
            GUILayout.Space (Spacing);
            GUILayout.EndVertical ();
            GUILayout.EndArea ();
            if (isPaused) {
            GUI.Box(new Rect(Screen.width/2-50/2,Screen.height/2-
12,100,25),"Paused");
            }
        } else {
            GUILayout.BeginArea (new Rect (20, 20, InitiateGUIWidth,
InitiateGUIHeight));
            GUILayout.BeginVertical ();
            GUILayout.BeginHorizontal ();
            GUILayout.BeginVertical ();
            GUILayout.Box ("Distance: " + Mathf.Round (distanceValue * 10) /
10 + " miles", GUILayout.Width (InitiateScrollWidth));
            distanceValue = GUILayout.HorizontalSlider (distanceValue,
distanceMin, distanceMax, GUILayout.Width (InitiateScrollWidth));
            GUILayout.Box ("Velocity: " + Mathf.Round (velocityValue * 10) /
10 + " mph", GUILayout.Width (InitiateScrollWidth));
            velocityValue = GUILayout.HorizontalSlider (velocityValue,
velocityMin, velocityMax, GUILayout.Width (InitiateScrollWidth));
```

```
            GUILayout.Box ("Acceleration: " + Mathf.Round (accelerationValue
* 10) / 10 + " mph/s", GUILayout.Width (InitiateScrollWidth));
            accelerationValue = GUILayout.HorizontalSlider
(accelerationValue, accelerationMin, accelerationMax, GUILayout.Width
(InitiateScrollWidth));
            GUILayout.EndVertical ();
            GUILayout.BeginVertical ();
            GUILayout.Box ("Select Vehicle", GUILayout.Width
(InitiateLabelWidth));
            //GUILayout.Space(15);
            GUILayout.BeginHorizontal ();
            if (GUILayout.Button ("<")) {
                //decrement Vehicle
                if (activeVehicle > 1) {
                    activeVehicle -= 1;
                } else {
                    activeVehicle = numVehicles;
                }
                updateVehicles ();
            }
            if (GUILayout.Button (">")) {
                //increment Vehicle
                if (activeVehicle < numVehicles) {
                    activeVehicle += 1;
                } else {
                    activeVehicle = 1;
                }
                updateVehicles ();
            }
            GUILayout.EndHorizontal ();
            GUILayout.Box (vehicleNames[activeVehicle - 1],
GUILayout.Width (InitiateLabelWidth));
            GUILayout.FlexibleSpace ();
            GUILayout.EndVertical ();
            GUILayout.BeginVertical ();
            write2file = GUILayout.Toggle (write2file, "Write Data to File",
GUILayout.Width (InitiateScrollWidth));
            if (write2file) {
                GUILayout.Space (10);
                GUILayout.Label ("Filename:");
                DataStorageFilenameString = GUILayout.TextField
(DataStorageFilenameString, GUILayout.Width (InitiateScrollWidth));
                GUILayout.Space (20);
            }
            //dispGraphs = GUILayout.Toggle (dispGraphs, "Display Graphs
(N/A Yet)");
            GUILayout.EndVertical ();
            GUILayout.EndHorizontal ();
            GUILayout.FlexibleSpace ();
            GUILayout.BeginHorizontal ();
            GUILayout.FlexibleSpace ();
            if (MatlabLoaded) {
                if (GUILayout.Button ("Run
Simulation",GUILayout.Width(150))) {
                    runningSim = true;
                    sendData ("Start_" + distanceValue + "_" +
velocityValue + "_" +
accelerationValue+"_"+dispGraphs+"_"+write2file+"_"+DataStorageFilenameString+"
_");
                }
            } else {
                //matlab isnt loaded
                if (Time.timeSinceLevelLoad <= 10f) {
                    GUILayout.Button ("Initalizing
Matlab...",GUILayout.Width(150));
                } else {
                    GUILayout.Button ("Matlab did not
Initalize",GUILayout.Width(150));
                }
            }
            if (GUILayout.Button ("Return to Lobby",GUILayout.Width(150)))
{
                sendExit ();
            }
            GUILayout.FlexibleSpace ();
            GUILayout.EndHorizontal ();
            GUILayout.EndVertical ();
            GUILayout.EndArea ();
        }
        if (endCondition){
            GUILayout.BeginArea(new Rect(Screen.width/2-
InitiateScrollWidth/2,Screen.height/2-200,InitiateScrollWidth,130));
            GUILayout.BeginVertical();
            if (GUILayout.Button("< Return")){
                sendRestart();
            }
            if (!write2file){
                GUILayout.FlexibleSpace ();
                GUILayout.Label ("Filename:");
                DataStorageFilenameString = GUILayout.TextField (DataStorageFilenameString, GUILayout.Width (InitiateScrollWidth));
                if (GUILayout.Button("Write Data to File")){
                    sendWrite2File();
                }
            }
            GUILayout.EndVertical();
            GUILayout.EndArea();
        }

    }

    void receiveData ()
    {
        while (mRunning) {
            try {

                localAddr = IPAddress.Parse ("127.0.0.1");
                tcp_Listener = new TcpListener (localAddr, port);
                tcp_Listener.Start ();
                //UnityEngine.Debug.Log("Server Started");
                client = tcp_Listener.AcceptTcpClient ();
                //UnityEngine.Debug.Log("Client Connected");

                ns = client.GetStream ();

                int i;
                Byte[] bytes = new Byte[256];
                String data = null;

                while ((i = ns.Read (bytes, 0, bytes.Length)) != 0) {
                    // Translate data bytes to a ASCII string.
                    data = System.Text.Encoding.ASCII.GetString (bytes, 0, i);
                    //UnityEngine.Debug.Log(data);

                    if (data.Equals ("Initalized\n")) {
                        MatlabLoaded = true;
                    } else if (data.Equals ("SimulationEnd\n")) {
                        endCondition = true;
```

```
					} else if (data.Equals ("restarting\n")) {
						restartScene();
					} else {
						parsedValues = data.Split ('_');
					}
				}

			} catch (ThreadAbortException e) {
				//UnityEngine.Debug.Log ("exception - possibly client
disconnected: " + e);
			} catch (Exception e) {
				//UnityEngine.Debug.Log ("Exception thrown: " + e);
			}
		}
		stopListening ();
	}

	void sendData (string Output)
	{
		encoder = new ASCIIEncoding ();
		OutputBuffer = encoder.GetBytes (Output);
		try {
			ns.Write (OutputBuffer, 0, OutputBuffer.Length);
		}
		catch {
			//UnityEngine.Debug.Log ("Connection to client terminated-
Message not sent");
		}
	}

	void OnApplicationQuit ()
	{
		sendExit ();
		// wait for listening thread to terminate (max. 500ms)
		//mThread.Join(500);
	}

	public void stopListening ()
	{
		mRunning = false;
		try {
			client.Close ();
		} catch {
```

```
                        UnityEngine.Debug.Log ("exception - possibly no client");
                }
                tcp_Listener.Stop ();
        }

        void sendRestart ()
        {
                sendData ("restart");
                FreeCamera.camera.enabled = false;
                VertCamera.camera.enabled = false;
                IsoCamera.camera.enabled = true;
        }

        void restartScene()
        {
                runningSim = false;
                PauseOrResume = "Run";
                isPaused = true;
                endCondition = false;
                write2file=false;
                parsedValues[0] = "0";
                parsedValues[1] = "0";
                parsedValues[2] = "0";
                parsedValues[3] = "0";
                parsedValues[4] = "0";
        }

        void sendPause ()
        {
                sendData ("Pause");
                PauseOrResume = "Resume";
                isPaused = true;
        }

        void sendResume ()
        {

                sendData ("Resume");
                PauseOrResume = "Pause";
                isPaused = false;
        }

        void sendExit ()
```

```
	{
		sendData ("exit");
		stopListening ();
		mThread.Join (500);
		try {
			matlab.Kill ();
		}
		catch{
		}


		Application.LoadLevel ("RoomLobby");
	}

	void sendWrite2File ()
	{
		sendData ("Write2File_" + DataStorageFilenameString +"_");
		write2file=true;
	}

	void updateVehicles ()
	{
		//disable all vehicle models
		Humvee.SetActiveRecursively (false);
		Car.SetActiveRecursively (false);
		jltv.SetActiveRecursively (false);
		M113.SetActiveRecursively (false);
		matv01_veh.SetActiveRecursively (false);
		Mrap.SetActiveRecursively (false);
		stryker.SetActiveRecursively (false);

		//import the working vehicle model and enable it

		if (activeVehicle == 1) {
			stryker.SetActiveRecursively (true);
		} else if (activeVehicle == 2) {
			matv01_veh.SetActiveRecursively (true);
		} else if (activeVehicle == 3) {
			Mrap.SetActiveRecursively (true);
		} else if (activeVehicle == 4) {
			M113.SetActiveRecursively (true);
		} else if (activeVehicle == 5) {
			Humvee.SetActiveRecursively (true);
			} else if (activeVehicle == 6) {
				jltv.SetActiveRecursively (true);
			} else {
				Car.SetActiveRecursively (true);
			}

			//set max velocity
			if (velocityValue > vehicleProperties[0, activeVehicle - 1]) {
				velocityValue = vehicleProperties[0, activeVehicle - 1];
			}
			velocityMax = vehicleProperties[0, activeVehicle - 1];

			//set max accel
			if (accelerationValue > vehicleProperties[1, activeVehicle - 1]) {
				accelerationValue = vehicleProperties[1, activeVehicle - 1];
			}
			accelerationMax = vehicleProperties[1, activeVehicle - 1];

			//set freelook camera position
			CameraAxes.transform.localPosition = new Vector3 (-3.139372f,
vehicleProperties[2, activeVehicle - 1], -2.776642f);
		}
}
```

APPENDIX G: MATLAB SCRIPT VEHICLE SIMULATION

```
%This listing outlines the MATLAB code (functions) used when setting the port for
TCP/IP communication between the Unity environment and MATLAB, as well as the
MATLAB code for data exchange between MATLAB and the main Unity application
(MATLAB is used as the driver for the Decision Support Center - Vehicle Simulation
application).

function [] = VehicleSimulation(port)

%http://www.peters-
research.com/index.php?option=com_content&view=article&id=61%3Aideal-lift-
kinematics&catid=3%3Apapers&Itemid=1

TCPLine = tcpip('127.0.0.1', port);
set(TCPLine, 'InputBufferSize', 30000);

fopen(TCPLine);
%send unity command to show matlab is initalized and ready to run program
fprintf(TCPLine, 'Initalized');

while (1)
   DistanceSimulation(TCPLine);
end

fclose(TCPLine);
delete(TCPLine);

end

function [] = writeData(filename,data)

if isempty(strfind(filename,'.txt'))
   filename=[filename '.txt'];
end

data(:,2)=data(:,2)/5280;
data(:,3:5)=3600/5280*data(:,3:5);
%headers = ['Time (sec)'; 'Distance (mi)'; 'Velocity (mph)'; 'Acceleration (mph/s)'; 'Jerk
(mph/s^2)'];

%dlmwrite(filename,headers,'delimeter', '\t','newline','pc');
dlmwrite(filename,data,'delimiter','\t','precision', 3,'newline','pc');

```

```
disp(['Data Writen to ' filename])

end

function [] = DistanceSimulation(TCPLine)

dt=1/60;

SplitArray={'null'};
isPause=1;
write2txt=0;

while ~strcmp(SplitArray{1},'Start')      %waits for input data to start with
   gotData=0;
   inputBuffer=[];
   while (gotData == 0)
      pause(.1);
      while (get(TCPLine, 'BytesAvailable') > 0)
         inputBuffer = [inputBuffer char(fread(TCPLine,1))];
         gotData = 1;
      end
      if gotData
         disp(inputBuffer);
         SplitArray=regexp(inputBuffer,'_','split');
         if strcmp(inputBuffer,'exit')
            exit;
         end
      end
   end
end

dispGraphs = strcmp(SplitArray{5},'True');
write2txt = strcmp(SplitArray{6},'True');
writeFilename = SplitArray{7};

D=str2num(SplitArray{2});      %miles
V=str2num(SplitArray{3});      %miles per hour
A=str2num(SplitArray{4});      %miles per hour per second
J=3;                           %miles per hour per second^2


%convert to stardard units for manipulation
D=D*5280;      %ft
```

```
V=V/3600*5280;    %ft per second
A=A/3600*5280;      %ft per second^2
J=J/3600*5280;            %ft per second^3

if (A/J>V/A)
   A=sqrt(V*J);
end

n=1;
if D >= (A^2*V+V^2*J)/(J*A)  %Reaches Full speed
   %disp('Reaches Max Speed');

   currJ=[0 J 0 -J 0 -J 0 J];

   tList(2)=A/J;
   tList(3)=V/A;
   tList(4)=A/J+V/A;
   tList(5)=D/V;
   tList(6)=D/V+A/J;
   tList(7)=D/V+V/A;
   tList(8)=D/V+A/J+V/A;
elseif D>=(2*A^3/J^2)  %Reaches Max Accel but not Max Speed
   %disp('Reaches Max Acceleration but Not Max Speed');

   currJ=[0 J 0 -J -J 0 J];

   tList(2)=A/J;
   tList(3)=-A/(2*J)+sqrt(A^3+4*D*J^2)/(2*J*sqrt(A));
   tList(4)=A/(2*J)+sqrt(A^3+4*D*J^2)/(2*J*sqrt(A));
   tList(5)=3*A/(2*J)+sqrt(A^3+4*D*J^2)/(2*J*sqrt(A));
   tList(6)=2*sqrt(A^3+4*D*J^2)/(2*J*sqrt(A));
   tList(7)=2*(A/(2*J)+sqrt(A^3+4*D*J^2)/(2*J*sqrt(A)));

else %D<2*A^3/J^2  %Doesnt Reach Max Speed of Accel
   %disp('Does not reach Max Acceleration or Speed');

   tList(2)=(D/(2*J))^(1/3);
   tList(3)=(4*D/J)^(1/3);
   tList(4)=(27/2*D/J)^(1/3);
   tList(5)=(32*D/J)^(1/3);

   currJ=[0 J -J -J J];
end
```

```
data=zeros(floor(tList(end)/dt),5);

currT=0;
currD=0;
currV=0;
currA=0;

for i=2:length(tList)
   t=0;
   TimeInterval=tList(i)-tList(i-1);
   while t < TimeInterval
      inputBuffer=[];
      while (get(TCPLine, 'BytesAvailable') > 0)
         inputBuffer = [inputBuffer char(fread(TCPLine,1))];
         gotData = 1;
      end
      if gotData
         disp(inputBuffer);
         if strcmp(inputBuffer,'Pause')
            isPause=1;
         elseif strcmp(inputBuffer,'exit')
            exit;
         elseif strcmp(inputBuffer,'restart')
            fprintf(TCPLine, 'restarting');
            return;
         end
      end
      if isPause
         gotData=0;
         inputBuffer=[];
         while (gotData == 0)
            pause(.05);
            while (get(TCPLine, 'BytesAvailable') > 0)
               inputBuffer = [inputBuffer char(fread(TCPLine,1))];
               gotData = 1;
            end
            if gotData
               disp(inputBuffer);
               if strcmp(inputBuffer,'Resume')
                  isPause=0;
               elseif strcmp(inputBuffer,'exit')
                  exit;
```

```
                elseif strcmp(inputBuffer,'restart')
                    fprintf(TCPLine, 'restarting');
                    return;
                end
            end
        end
    end

    t=t+dt;
    n=n+1;
    data(n,1)=t+currT;
    data(n,2)=currD+currV*t+currA*t^2/2+currJ(i)*t^3/6;
    data(n,3)=currV+currA*t+currJ(i)*t^2/2;
    data(n,4)=currA+currJ(i)*t;
    data(n,5)=currJ(i);
                 %distance in ft for travel

tcpPacket=strcat(num2str(round(data(n,2)*1000)/1000),'_',num2str(round(data(n,1)*
10)/10),'_',num2str(round(data(n,2)/5.28)/1000),'_',num2str(round(data(n,3)*150/22
)/10),'_',num2str(round(data(n,4)*150/22)/10));
    pause(.01);
    fprintf(TCPLine,tcpPacket);
  end
  currT=data(n,1);
  currD=data(n,2);
  currV=data(n,3);
  currA=data(n,4);
end
tcpPacket='SimulationEnd';
fprintf(TCPLine,tcpPacket);

if write2txt
  writeData(writeFilename,data);
end

while (1)
  gotData=0;
  inputBuffer=[];
  while (gotData == 0)
    pause(.05);
    while (get(TCPLine, 'BytesAvailable') > 0)
      inputBuffer = [inputBuffer char(fread(TCPLine,1))];
      gotData = 1;
```

```
        end
        if gotData
           disp(inputBuffer);
           SplitArray=regexp(inputBuffer,'_','split');
           if strcmp(inputBuffer,'exit')
              exit;
           elseif strcmp(inputBuffer,'restart')
              fprintf(TCPLine, 'restarting');
              return;
           elseif strcmp(SplitArray{1},'Write2File')
              writeFilename=SplitArray{2};
              writeData(writeFilename,data);
           end
        end
     end
end

% %data(:,2)=data(:,2)/5280;
% %data(:,3:5)=3600/5280*data(:,3:5);
%
%
% %sp(1)=subplot(3,1,1);plot(data(:,1),data(:,2))
% %sp(2)=subplot(3,1,2);plot(data(:,1),data(:,3))
% %sp(3)=subplot(3,1,3);plot(data(:,1),data(:,4))
% %linkaxes(sp,'x');
% %ylabel(sp(1),'Position [mi]');ylabel(sp(2),'Velocity [mph]');ylabel(sp(3),'Acceleration
[mph/s]');xlabel(sp(3),'Time [s]')

End
```